End of Master Project

Master of Industrial Engineering

Analysis of feasibility in OpenSim Moco of Inverse Dynamics problems involving human models with exoskeletons

Author: Guillermo Muñoz Pérez

Tutors: Juana Mª Mayo Núñez

Joaquín Ojeda Granja

Mechanical Engineering and Manufacturing Department
Higher Technical School of Engineering
University of Seville

Seville, 2021

End of Master Project
Master of Industrial Engineering

Analysis of feasibility in OpenSim Moco of Inverse Dynamics problems involving human models with exoskeletons

Author:
Guillermo Muñoz Pérez

Tutores:
Juana Mayo Núñez
Joaquín Ojeda Granja

Dpto. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

End of Master Project: Analysis of feasibility in OpenSim Moco of Inverse Dynamics problems involving human models with exoskeletons

Autor: Guillermo Muñoz Pérez

Tutores: Juana Mª Mayo Núñez

Joaquin Ojeda Granja

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

Agradecimientos

Tras tantos meses de duro trabajo, videoconferencias, horas frente al ordenador, lágrimas de agobio y desesperación, y momentos de alegría tras cada pequeño logro conseguido, este Trabajo llega a su fin, y con él mi etapa académica, que no formativa, pues uno siempre se está formando y aprendiendo en este camino que es la vida.

Y mucho he aprendido en estos más de 20 años. Empezando por la señorita Antoñita (descanse en paz), de Educación Infantil, que me enseñó a contar y a escribir las letras y los números en un papel. Siguiendo por las valiosas lecciones de esos Maestros y Profesores que dignifican el nombre de su profesión, y no puedo evitar homenajear a mi recordado Maestro Don Fausto (descanse en paz). Allá donde esté, este Trabajo va dedicado a usted. Y concluyendo con los valiosos consejos que me han aportado mis Tutores en este Trabajo, Juana y Joaquín, de los que tanto he aprendido y que trataré siempre de poner en práctica en mi presente y futuro personal y profesional.

Pero por supuesto, debo citar a mi familia. Esa familia que me ha criado, cuidado, educado y formado como persona. Esos padres, Frank y Maricarmen, que se han desvivido por mí y por mis hermanos, y esos hermanos, Javi y Carmen, que siempre están ahí para lo bueno y para lo malo. Y por último, pero no menos importante, sino todo lo contrario. La familia que he elegido, la persona que me ha acompañado desde el principio hasta el fin de este Trabajo, tragándose todos mis lamentos y agobios, apoyándome en todo momento, aportándome los consejos más valiosos del mundo y alegrándose de mis éxitos y avances tanto como de los suyos. Azahara, amor mío, este Trabajo, y la vida en general, serían mucho más difíciles sin ti. Te quiero.

Guillermo Muñoz Pérez

Sevilla, 2021

Resumen

En este Trabajo se analiza la factibilidad a la hora de realizar en OpenSim Moco análisis dinámicos inversos en modelos humano-exoesqueleto. Estos modelos son mecanismos de cadena cerrada con músculos, por lo que el problema puede reducirse al estudio de la Dinámica Dnversa de un mecanismo de 4 barras con músculos. En el Capítulo 1, se muestra una concisa introducción sobre los exoesqueletos, así como una breve reseña bibliográfica sobre el estudio de la Dinámica Inversa en el cuerpo humano, especialmente en los casos con exoesqueletos.

El Capítulo 2 contiene una introducción de OpenSim Moco, en la cual se desarrolla el problema del control óptimo, las técnicas de colocación directa empleadas por Moco para resolver el problema, y las diferentes problemáticas que se pueden resolver en Moco, con especial atención al MocoInverse, resuelto en este Proyecto. A continuación, el Capítulo 3 versa sobre los modelos musculares empleados en este Trabajo, describiéndose las ecuaciones que emplean para obtener las fuerzas musculares, así como sus curvas características.

Más adelante, el Capítulo 4, se detalla cómo aborda OpenSim el problema del bucle abierto, para después introducir la problemática de los mecanismos de bucle cerrado, el tema central de este Trabajo, y cómo son resueltos. La formulación genérica de la Cinemática y Dinámica Inversa para sistemas multicuerpo es presentada también. El Capítulo 5 supone la aparición del mecanismo de cuatro barras (primero sin músculos y después con ellos), además de la particularización de los problemas presentados con anterioridad para el cuadrilátero articulado.

El Capítulo 6 sirve para mostrar los resultados del Trabajo, demostrando la hipótesis que había servido de punto de partida para esta investigación: la factibilidad de usar OpenSim Moco para resolver la Dinámica Inversa del conjunto humano-exoesqueleto. También se muestra un pequeño análisis de sensibilidad acerca de la influencia de los cambios en el punto de inserción de los músculos, así como del número de músculos, en el comportamiento dinámico del sistema. Por último, una breve conclusión sobre el Proyecto cierra esta investigación. En el Anexo final aparecen algunos resultados intermedios, sin importancia directa en los resultados del Trabajo, pero que pueden resultar de interés para el lector.

Palabras clave: Dinámica Inversa, Cuatro Barras, Exoesqueleto, Activación/Contracción Muscular

Abstract

In this work, the feasibility to perform Inverse Dynamics problems in human models wearing exoskeletons by using OpenSim Moco is studied. These models are closed-loop mechanisms with muscles, so the problem can be reduced to study Inverse Dynamics in four-bars mechanisms with muscles. In Chapter 1, a brief introduction about exoskeletons is shown, as well as a bibliographic revision about Inverse Dynamics applied to the human body, specially when wearing exoskeletons.

In Chapter 2, OpenSim Moco is introduced, developing the optimal control problem, the direct collocation methods to solve it, and the diverse types of problems which can be found in Moco, focusing on MocoInverse (the one used in this report). Then, in Chapter 3, muscle models are analyzed in terms of characteristic curves and equations to obtain muscle force.

Next, in Chapter 4, the way OpenSim solves open-loop problems is detailed, to then introduce the issue of closed-chain mechanisms, and how they are faced in this work. Moreover, the theoretical expressions of Inverse Kinematics and Dynamics for multibody systems are presented. In Chapter 5, the 4-bar mechanism appears (firstly without muscles, and then, with them), as well as the particularization of every problem previously explained on it.

Chapter 6 serves to display every result obtained, demonstrating the feasibility of the human-exoskeleton Inverse Dynamics in OpenSim Moco, and also showing a little analysis of sensitivity about the influence of changing the insertion point of muscles and the number of them. By last, a brief conclusion about the work closes the research. In the Annex, some intermediate results involving muscle forces and parameters are displayed.

Keywords: Inverse Dynamics, Four Bar Mechanism, Exoskeleton, Muscle Contraction/Activation

Index

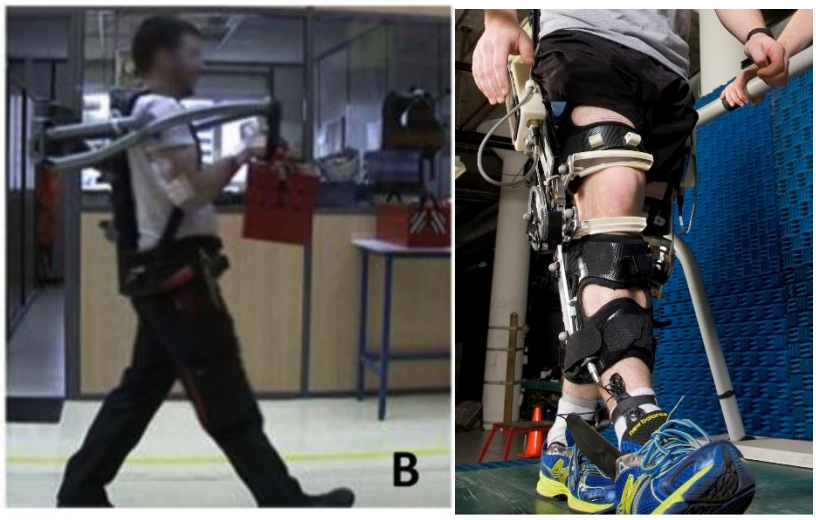
LIST OF TABLES

LIST OF FIGURES

Figure 3.6. Active force length curve. OpenSim Documentation [28] 24
Figure 3.7. Tendon force length curve. OpenSim Documentation [28] 25
Figure 3.8. Tendon force-strain curve by Thelen. Obtained in Matlab 26
Figure 3.9. Tendon force-strain curve by Thelen. Obtained in Matlab 27
Figure 3.10. Active force-length curve by Thelen. Obtained in Matlab 27
Figure 3.11. Force velocity curve by Thelen. Obtained in Matlab 28

1 INTRODUCTION

The implementation of exoskeletons in human people is one of the most complex and exciting challenges being faced by Biomechanics since the last few decades. These devices, whose origin can be found on the military field [2], help people to perform diverse types of motion by supporting an extra amount of energy. There are two types of exoskeletons according to the source of energy [1]: active exoskeletons, if they contain an electrical device which supports energy from a battery or other sort of external source like pneumatical or hydraulic circuits; and passive exoskeletons, whose way to obtain energy consists of the springs, dampers and other elastic elements which store energy and release it during the motion. The biomechanical models used in this work simulate passive exoskeletons, because no electrical actuator is implemented.



Figures 1.1 and 1.2: Industrial worker wearing an Exhauss passive exoskeleton (Theurel et al., [3]) and an active exoskeleton worn in lower limbs (Yale University, [4])

The study of dynamics in the whole human-exoskeleton is not an easy problem to solve, as two complex systems are being considered, in the sense of the great magnitude of bodies, joints and muscles they present. Moreover, the only form to attach an exoskeleton is by linking it to the person through two points, creating therefore a closed loop which complicates even more the problem.

This project has the goal to evaluate the adequacy of OpenSim Moco (Stanford University, [5]) to solve Inverse Dynamics problems (it is said, problems where forces and moments are calculated parting from a given and perfectly defined kinematics) in musculoskeletal models with exoskeletons by minimizing a determined cost function, it is said, solving an optimal control problem where the equations of motion are part of the constraints in the system. OpenSim Moco is a software developed by Stanford University [5] which solves optimal control problems by direct collocation methods that will be detailed in Chapter 3. Moco is part of OpenSim, an open-source software system also powered by Stanford University [6], which serves to create musculoskeletal models and perform simulations with them. Every model in this work have been built with OpenSim.

The difficult to study such a complex closed-loop model in OpenSim like a human wearing an exoskeleton has

led to work with a simpler model, but also valid, since it is still a closed-chain device: a muscled 4-bar mechanism. So, the objective of this project, just defined, will be fulfilled if an Inverse Dynamics analysis is successfully performed in this muscled 4-bar linkage.

Diverse research has been done about Inverse Dynamics problems in human people. Bueno and Montano [7] studied how to obtain joint torques in upper limbs parting from s-EMG (electromyographical) information, while Sartori, Reggiani, Lloyd and Pagello make the same in [8], but considering lower limbs. In the University of Sevilla, Maza [9] implemented an Inverse Dynamics problem in OpenSim and Matlab, applied to human walking. Studies with exoskeletons are more complex, but there exist some examples in the literature such as the work of García Vallejo, Font-Llagunes and Schliehen [10], who presented the design of an active orthosis to solve a dynamic problem through aesthetic and energetic optimization.

Concerning the optimal control problem applied to musculoskeletal models, the work by Rina García [11] must be noted thanks to her development of an optimal control methodology to predict human movement.

Regarding methods to solve optimal control problems, the single shooting problem must be cited. It minimizes an objective function which contains a control signal by comparing the error between the reached value of the state variable and the required value it must have. It is less optimal than direct collocation methods it does not take in count the constraints in the objective function. Direct collocation methods will be further explained in Chapter 2.

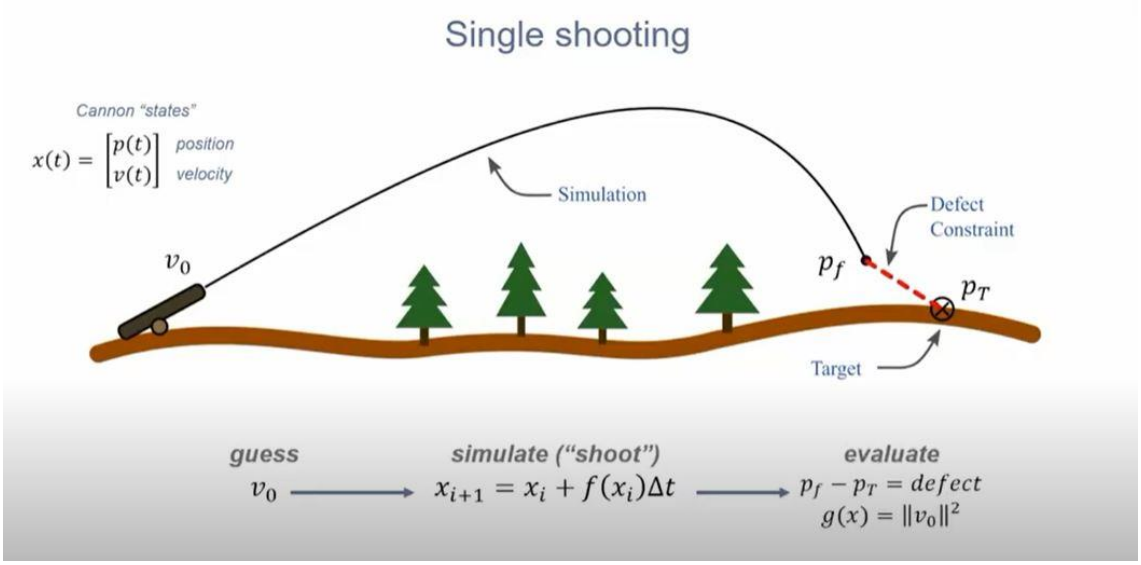


Figure 1.3. Scheme of single shooting problem. [12]

2 OpenSim Moco

In this chapter, OpenSim Moco, the tool utilized to face and solve optimal control problems, will be presented. First of all, a general introduction about Moco software will be done, including its utilities and what sorts of problems it can approach. Then, the optimal control problem will be slightly explained and detailed, as it involves every tipologies of problems that can be tackled. Hereafter, direct collocation methods will be described as a way used by Moco to solve the optimal control problem by turning it into a non-linear discrete problem. Last, the different types of problems available in Moco will be detailed.

2.1 What is Moco?

The word Moco is the acronym for Musculoskeletal Optimal Control, being these words explanatory enough about Moco's utility. This tool solves optimization problems where control variables are minimized so as to obtain certain magnitudes related to musculoskeletal activity in an OpenSim model [13]. These magnitudes can be the forces which generate a given movement (Inverse Dynamics), muscle activity, mass properties… it depends on the sort of problem faced.

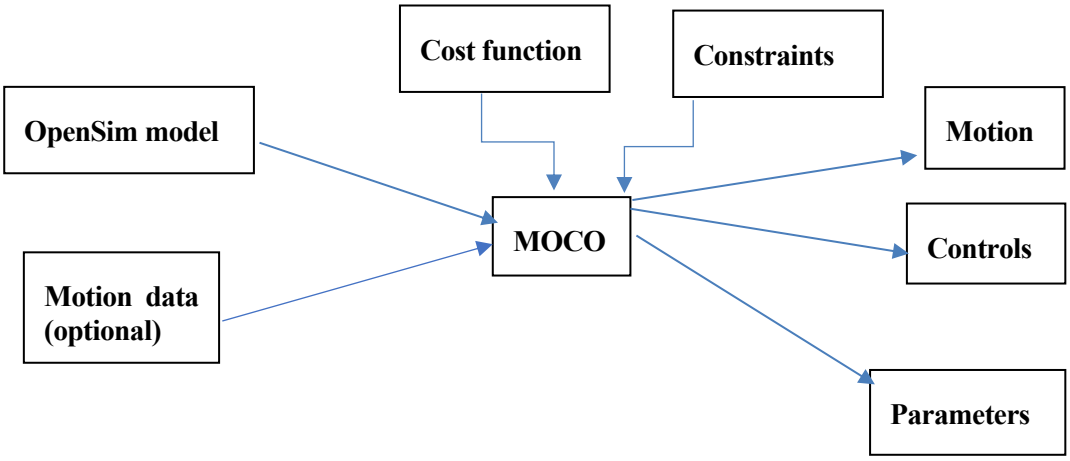


Figure 2.1. Operating scheme of OpenSim Moco. [13]

The scheme above briefly shows how Moco works, what inputs it must receive, how this tool must be configurated, and what results can produce. Regarding the inputs, Moco always works with an OpenSim model, which will include all information about its mass properties, its muscles, as well as the kinematic constraints. Additionally, depending on the type of problem, motion data can be introduced. If this motion is prescribed, an Inverse Dynamic problem (MocoInverse) [13] will be solved. If there is information about the motion but it is not certainly defined, a tracking problem (MocoTracking) [13] will be the one to solve. If there is no information about motion, a generic MocoStudy problem [13] will be faced. In subchapter 3.4 these possible problems will be further described.

2.2 The optimal control problem

The mathematical problem faced in Moco is the optimal control problem, which deals with finding those control variables that, while fulfilling certain dynamical restrictions, are able to minimize as much as possible an objective function (also named cost function) [14]. The most generic cost function in Moco is the following one [13]:

$$\min \sum_j w_j J_j(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{c,j}) \quad \textbf{(2.1)}$$

Where J_j corresponds to the j-th cost term, w_j is its associated weight, t_0 and t_f the initial and the final instant of time, y is a vector containing generalized coordinates and speed through time, x is a vector containing time histories of control variables, λ is the vector with kinematic constraints multipliers, p is a parameter which works as an exponent, and $S_{c,j}$ consists on the integral of the j-th cost goal, $s_{c,j}$:

$$S_{c,j} = \int_{t_0}^{t_f} s_{c,j}(t, y, x, \lambda, p)dt \quad \textbf{(2.2)}$$

Control variables are those whose function is to regulate the behavior of the system by minimizing the objective function, while state variables are those which describe the evolution of the system [14]. As just seen, some auxiliary state variables may appear in the objective function; these can be either the muscle activation or the muscle fiber length.

Once defined the cost function to minimize, constraints must be described. As every dynamical system, the problem in Moco is regulated by the equations of motion, which result to be the most important constraints in the system [15]:

$$u = \dot{q} \quad \textbf{(2.3)}$$

$$M(q,p) * \dot{u} + G(q,p)^T * \lambda = f_{app}(t, y, x, p) - f_{inertial}(q, u, p) \quad \textbf{(2.4)}$$

Where u is the second derivative of the generalized coordinates, $M(q,p)$ is the mass matrix, G^T is the transpose jacobian matrix of constraints, λ is the vector containing Lagrange multipliers, f_{app} is the vector of applied forces (gravity and muscle forces particularly), and $f_{inertial}$ is the vector containing inertial forces (centripetal, giroscopic and Coriolis forces).

If auxiliary state variables exist, they follow their own dynamic equations, which can be explicit, if their first derivative is cleared (Equation 2.5), or implicit if this first derivative belongs to a term equaled to zero (Equation 2.6):

$$\dot{z}_{ex}(t) = f_{\dot{z}_{ex}(t)}(t, y, x, \lambda, p) \quad \textbf{(2.5)}$$

$$0 = f_{\dot{z}_{im}(t)}(t, y, \dot{z}_{im}(t), x, \lambda, p) \quad \textbf{(2.6)}$$

In Chapter 3 the equation relating the time derivative of activation and muscle fiber length will be introduced.

Next, kinematic constraints must be defined. These determine the relations between generalized coordinates, and are expressed are equations containing a combination of these coordinates equaled to zero [13]:

$$\Phi(q,p) = 0 \quad \textbf{(2.7)}$$

Another set of constraints to be defined are the boundaries, which impose upper and lower limits to determined expressions involving every variable in the problem [15]:

$$V_{L,k} \leq V_k(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{b,k}) \leq V_{U,k} \quad \textbf{(2.8)}$$

Where $S_{b,k}$ is the integral through time of $s_{b,k}$, a term belonging to the k_{th} boundary constraint goal:

$$S_{b,k} = \int_{t_0}^{t_f} s_{b,k}(t, y, x, \lambda, p)dt \quad \textbf{(2.9)}$$

Last, path constraints may be described as those which enforce constraints along the trajectory [15]:

$$g_L \leq (t, y, x, \lambda, p) \leq g_U \quad \textbf{(2.10)}$$

Every constraint has already been described, so it is the moment to define boundaries associated to every variables involved in the problem. Starting with limits imposed to initial and final values of states and controls [13]:

$$y_{0,L} \leq y_0 \leq y_{0,U} \quad \textbf{(2.11)}$$

$$y_{f,L} \leq y_f \leq y_{f,U} \quad \textbf{(2.12)}$$

$$x_{0,L} \leq x_0 \leq x_{0,U} \quad \textbf{(2.13)}$$

$$x_{f,L} \leq x_f \leq x_{f,U} \quad \textbf{(2.14)}$$

Continuing with the initial and final instant of time:

$$t_{0,L} \leq t_0 \leq t_{0,U} \quad \textbf{(2.15)}$$

$$t_{f,L} \leq t_f \leq t_{f,U} \quad \textbf{(2.16)}$$

Following with the states and controls themselves:

$$y_L \leq y(t) \leq y_U \quad \textbf{(2.17)}$$

$$x_L \leq x(t) \leq x_U \quad \textbf{(2.18)}$$

And finishing with time-invariant parameter p:

$$p_L \leq p \leq p_U \quad \textbf{(2.19)}$$

Now that the cost function, the constraints, the boundaries and variables have been presented, all terms can be expressed in the generic formulation of the optimal control problem faced by Moco [13]:

$$\min \sum_j w_j J_j(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{c,j})$$

$$S_{c,j} = \int_{t_0}^{t_f} s_{c,j}(t, y, x, \lambda, p)dt$$

subject to

$$u = \dot{q}$$

$$M(q,p) * \dot{u} + G(q,p)^T * \lambda = f_{app}(t, y, x, p) - f_{inertial}(q, u, p)$$

$$\dot{z}_{ex}(t) = f_{\dot{z}_{ex}(t)}(t, y, x, \lambda, p)$$

$$0 = f_{\dot{z}_{im}(t)}(t, y, \dot{z}_{im}(t), x, \lambda, p)$$

$$\Phi(q,p) = 0$$

$$V_{L,k} \leq V_k(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{b,k}) \leq V_{U,k}$$

$$S_{b,k} = \int_{t_0}^{t_f} s_{b,k}(t, y, x, \lambda, p)dt$$

$$g_L \leq g(t, y, x, \lambda, p) \leq g_U$$

with respect to

$$y_{0,L} \leq y_0 \leq y_{0,U}$$

$$y_{f,L} \leq y_f \leq y_{f,U}$$

$$x_{0,L} \leq x_0 \leq x_{0,U}$$

$$x_{f,L} \leq x_f \leq x_{f,U}$$

$$t_{0,L} \leq t_0 \leq t_{0,U}$$

$$t_{f,L} \leq t_f \leq t_{f,U}$$

$$y_L \leq y(t) \leq y_U$$

$$x_L \leq x(t) \leq x_U$$

$$p_L \leq p \leq p_U$$

$$\lambda(t)$$

Now it is time to describe the methods used by Moco to solve the optimal control problem: direct collocation methods.

2.3 Direct collocation methods

The problem just presented, the optimal control, is a trajectory optimization problem continuous in time, as it introduces integrals in the objective function. To solve it, Moco makes use of direct collocation methods [13]. These algorithms transform the optimal control problem into a nonlinear discrete optimization problem, by discretizing the trajectory optimization problem into a determined number of control points, named nodes [15].

The continuous functions belonging to the original optimal control problem are substituted by polynomial splines whose value in control points must be the same as in the continuous functions. A graphic representation is shown in Figure 2.2:

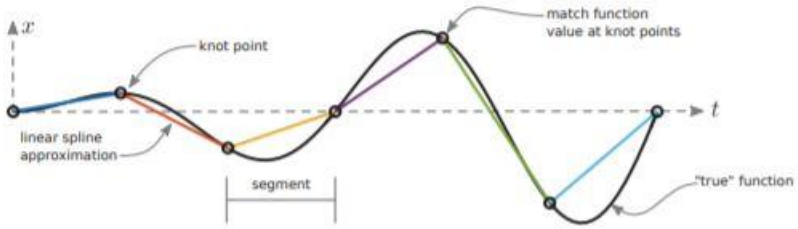


Figure 2.2. Lineal interpolation through nodes in optimal control problem. Kelly, M. [15]

One of the main advantages about using direct collocation resides on the fact that the dynamic constraints (it is said, the equations of motion), are taken in count in the cost function, as they include control variables which must minimize the goal [12, 15]. This makes slightly easier to find a solution for the optimization problem.

Moco can use two different collocation methods to solve the optimal control problem: the trapezoidal method, and the Hermite-Simpson's method [13].

2.3.1 Trapezoidal collocation

The trapezoidal direct collocation method uses the trapezoidal quadrature rule to discretize the continuous problem in N control points and approximate the integrals as summations of discrete variables, evaluated in those control nodes [15]:

$$\int_{t_0}^{t_f} f\big(\tau, x(\tau)\big)\, d\tau \approx \sum_{i=0}^{N-1} trap_i = \sum_{i=0}^{N-1} \frac{1}{2} * (t_{i+1} - t_i) * (f_i + f_{i+1}) \qquad \textbf{(2.20)}$$

Where N is the total number of control points and $f\big(\tau, x(\tau)\big)$ is any function belonging to an integrand in the optimization problem. It is needed that the following condition gets respected in each control point [15]:

$$f(t_i) = \widetilde{f}_i \quad \textbf{(2.21)}$$

It is said, in control points, the value of the continuous function and its approximative spline must be the same.

The process to obtain the values of control and state variables can be explained as follows in the next lines. In first place, the spline used to approximate the control trajectories is a linear polynomial [15]:

$$u(t) \approx u_k + \frac{\tau}{h_k} * (u_k + u_{k+1}) \quad \textbf{(2.22)}$$

Where τ and h_k are expressed as follows:

$$\tau = t - t_k \quad \textbf{(2.23)}$$

$$h_k = t_{k+1} - t_k \quad \textbf{(2.24)}$$

Regarding states, the dynamic equations can be represented in function of the first derivative of the state variables:

$$\dot{x} = f \quad \textbf{(2.25)}$$

If trapezoidal quadrature rule is used, the following relation is obtained [15]:

$$f \approx f_k + \frac{\tau}{h_k} * (f_k + f_{k+1}) \quad \textbf{(2.26)}$$

State trajectories are obtained by integrating this equation:

$$x \approx x_k + f_k * t + \frac{\tau^2}{2h_k} * (f_k + f_{k+1}) \quad \textbf{(2.27)}$$

This way, after applying the trapezoidal rule, the following non-linear discrete optimization problem appears [13]:

$$\min \sum_j w_j J_j(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{c,j}) + w_\lambda * \sum_{i=1}^{n} trap_i(\|\lambda_i\|_2^2)$$

$$S_{c,j} = \sum_{i=0}^{N} trap_i(s_{c,j}(t, y, x, \lambda, p))$$

subject to

$$u_i = u_{i-1} + trap_i(v) \quad i = 1, \dots, N$$

$$M(q_i, p) * v_i + G(q, p)^T * \lambda_i = f_{app}(t_i, y_i, x_i, p) - f_{inertial}(q_i, u_i, p) \qquad i = 0, \dots, N$$

$$z_{ex_i} = z_{ex,i-1} + trap_i\left(f_{ex,\dot{z}}(t, y, x, \lambda, p)\right) \quad i = 1, \dots, N$$

$$z_{im_i} = z_{im,i-1} + trap_i(\zeta) \quad i = 1, \dots, N$$

$$0 = f_{\dot{z}_{im}(t)}(t_i, y_i, \zeta_i, x_i, \lambda_i, p) \qquad i = 0, \dots, N$$

$$\Phi(q_i, p) = 0 \qquad i = 0, \dots, N$$

$$V_{L,k} \le V_k(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{b,k}) \le V_{U,k}$$

$$S_{b,k} = \sum_{i=0}^{N} trap_i(s_{b,k}(t, y, x, \lambda, p)) \qquad k = 1, \dots, K$$

$$g_L \le g\left(t_i, y_{_i}, x_i, \lambda_i, p\right) \le g_U \qquad i = 1, \dots, N$$

with respect to

$$v_i \in [-v_B, v_B]$$

$$y_{0,L} \le y_0 \le y_{0,U}$$

$$y_{f,L} \le y_n \le y_{f,U}$$

$$y_L \le y_i \le y_U \quad i = 0, \dots, N-1$$

$$x_{0,L} \le x_0 \le x_{0,U}$$

$$x_{f,L} \le x_n \le x_{f,U}$$

$$x_L \le x_i \le x_U \quad i = 0, \dots, N-1$$

$$t_{0,L} \le t_0 \le t_{0,U}$$

$$t_{f,L} \le t_n \le t_{f,U}$$

$$y_L \le y(t) \le y_U$$

$$x_L \le x(t) \le x_U$$

$$p_L \le p \le p_U$$

$$\zeta_L \le \zeta_i \le \zeta_U \qquad i = 0, \dots, N$$

$$\lambda_L \le \lambda_i \le \lambda_U \qquad i = 0, \dots, N$$

Where v is generalized acceleration and v_B a very large quantity [13].

2.3.2 Hermite-Simpson collocation

This algorithm is the one used by default by Moco. It allows more accurate results than the trapezoidal rule because it introduces additional collocation points at the mesh interval midpoints, resulting to a total of 2N+1 grid points in which the trajectory is discretized [15]. This way, control trajectories are approximated by quadratic splines, while state trajectories are approximated to cubic splines.

First of all, it is necessary to define the Simpson's rule of quadrature [15]:

$$\int_{t_0}^{t_f} f\big(\tau, x(\tau)\big)\, d\tau \approx \sum_{i=0}^{N-1} simpson_i = \sum_{i=0}^{N-1} \frac{h_i}{6} * (f_{i-1} + 4 * f_i + f_{i+1}) \quad \textbf{(2.28)}$$

With h_i defined as in Equation. The value of the state variable in the midpoint belonging to the instant $t_{k+1/2}$ is obtained with the Hermite interpolant:

$$hermite_{k+1/2} = x_{k+\frac{1}{2}} = \frac{1}{2} * (x_k + x_{k+1}) + \frac{h_k}{8} * (f_k - f_{k+1}) \quad \textbf{(2.29)}$$

Following equations to obtain the control and state variables through time are demonstrated in Kelly, M. [15]:

$$u(t) = \frac{2}{h_k^2} * \left(\tau - \frac{h_k}{2}\right) * (\tau - h_k) * u_k - \frac{4}{h_k^2} * (\tau) * (\tau - h_k) * u_{k+\frac{1}{2}} + \frac{2}{h_k^2} * (\tau) * (\tau - h_k) * u_{k+1} \quad \textbf{(2.30)}$$

$$x(t) = x_k + f_k\left(\frac{\tau}{h_k}\right) + \frac{1}{2} * \left(-3f_k + 4f_{k+\frac{1}{2}} - f_{k+1}\right) * \left(\frac{\tau}{h_k}\right)^2 + \frac{1}{3} * \left(2f_k - 4f_{k+\frac{1}{2}} + 2f_{k+1}\right) * \left(\frac{\tau}{h_k}\right)^3 \quad \textbf{(2.31)}$$

Therefore, the formulation of the optimal control problem after applying the Hermite-Simpson collocation method would be the following one [13]:

$$\min \sum_j w_j J_j(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{c,j}) + w_\lambda * \sum_{i=1}^{n} simpson_i(\|\lambda_i\|_2^2)$$

$$S_{c,j} = \sum_{i=0}^{N} simpson_i(s_{c,j}(t, y, x, \lambda, p))$$

subject to

$$\bar{u}_i = hermite_i(u, v) \qquad i = 1 \dots N$$
$$u_i = u_{i-1} + simpson_i(v) \qquad i = 1 \dots N$$

$$M(q_i, p) * v_i + G(q_i, p)^T * \lambda_i = f_{app}(t_i, y_i, x_i, p) - f_{inertial}(q_i, u_i, p) \qquad i = 0, \dots, N$$
$$M(\bar{q}_i, p) * v_i + G(\bar{q}_i, p)^T * \lambda_i = f_{app}(\bar{t}_i, \bar{y}_i, \bar{x}_i, p) - f_{inertial}(\bar{q}_i, \bar{u}_i, p) \qquad i = 1, \dots, N$$
$$z_{ex_i} = z_{ex,i-1} + simpson_i\left(f_{ex,\dot{z}}(t, y, x, \lambda, p)\right) \qquad i = 1, \dots, N$$
$$\bar{z}_{ex_i} = hermite_i\left(f_{ex,\dot{z}}(t, y, x, \lambda, p)\right) \qquad i = 1, \dots, N$$
$$\bar{z}_{im_i} = hermite_i(\zeta) \qquad i = 1, \dots, N$$
$$z_{im_i} = z_{im,i-1} + simpson_i(\zeta) \qquad i = 1, \dots, N$$
$$0 = f_{\dot{z}_{im}(t)}(t_i, y_i, \zeta_i, x_i, \lambda_i, p) \qquad i = 0, \dots, N$$
$$\bar{x}_i = \frac{x_{i-1} + x_i}{2} \qquad i = 1, \dots, N$$
$$\Phi(q_i, p) = 0 \qquad i = 0, \dots, N$$
$$\dot{\Phi}(q_i, u_i, p) = 0 \qquad i = 0, \dots, N$$
$$\ddot{\Phi}(t_i, y_i, x_i, \lambda_i, p) = 0 \qquad i = 0, \dots, N$$

$$V_{L,k} \le V_k(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{b,k}) \le V_{U,k}$$
$$S_{b,k} = \sum_{i=0}^{N} simpson_i(s_{b,k}(t, y, x, \lambda, p)) \qquad k = 1, \dots, K$$
$$g_L \le g\left(t_i, y_i, x_i, \lambda_i, p\right) \le g_U \qquad i = 1, \dots, N$$

with respect to

$$y_{0,L} \le y_0 \le y_{0,U}$$
$$y_{f,L} \le y_n \le y_{f,U}$$
$$y_L \le y_i \le y_U \quad i = 0, \dots, N-1$$
$$y_L \le \bar{y}_i \le y_U \quad i = 0, \dots, N$$

$$x_{0,L} \le x_0 \le x_{0,U}$$
$$x_{f,L} \le x_n \le x_{f,U}$$
$$x_L \le x_i \le x_U \quad i = 0, \dots, N-1$$

$$x_L \leq \bar{x}_i \leq x_U \quad i = 0, \dots, N$$

$$t_{0,L} \leq t_0 \leq t_{0,U}$$

$$t_{f,L} \leq t_n \leq t_{f,U}$$

$$p_L \leq p \leq p_U$$

$$\zeta_L \leq \zeta_i \leq \zeta_U \qquad i = 0, \dots, N$$

$$\zeta_L \leq \bar{\zeta}_i \leq \zeta_U \qquad i = 0, \dots, N$$

$$\lambda_L \leq \lambda_i \leq \lambda_U \qquad i = 0, \dots, N$$

$$\lambda_L \leq \bar{\lambda}_i \leq \lambda_U \qquad i = 0, \dots, N$$

2.4 Types of Moco problems

Moco can solve fundamentally three types of problems, depending on the presence or absence of information about the model's motion [12, 13].

2.4.1 MocoInverse

This is the type of problem faced in this work. In MocoInverse, model's kinematics is completely prescribed, so the objective is to obtain the forces which produce this motion, via the control variables present in the objective function, and the actuator states [12, 13]. Therefore, it deals with an Inverse Dynamic problem. The scheme in Figure helps to understand it:

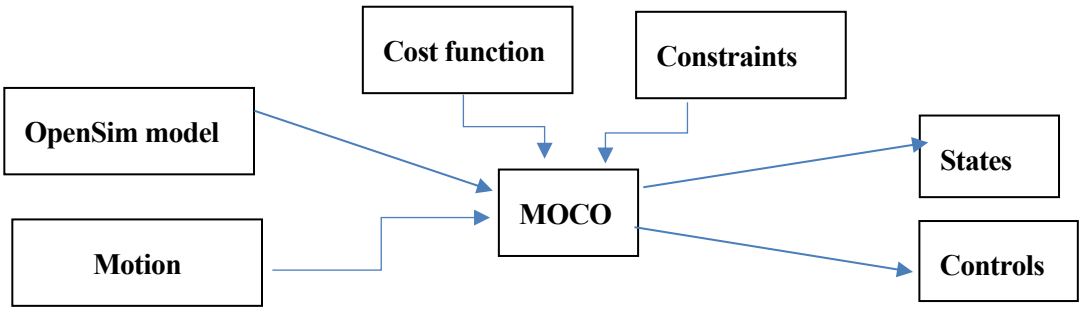


Figure 2.3. Operating scheme of MocoInverse problems

This is the type of problem approached in this work, so it is useful to know the formulation of the problem. It is presented right below:

$$\min \sum_j w_j J_j(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{c,j})$$

$$S_{c,j} = \int_{t_0}^{t_f} s_{c,j}(t, y, x, \lambda, p) dt$$

subject to

$$u = \dot{q}$$

$$M(q,p) * \dot{u} + G(q,p)^T * \lambda = f_{app}(t, y, x, p) - f_{inertial}(q, u, p)$$

$$\dot{z}_{ex}(t) = f_{\dot{z}_{ex}(t)}(t, y, x, \lambda, p)$$

$$0 = f_{\dot{z}_{im}(t)}(t, y, \dot{z}_{im}(t), x, \lambda, p)$$

$$V_{L,k} \leq V_k(t_0, t_f, y_0, y_f, \lambda_0, \lambda_f, x_0, x_f, p, S_{b,k}) \leq V_{U,k}$$

$$S_{b,k} = \int_{t_0}^{t_f} s_{b,k}(t, y, x, \lambda, p) dt \qquad k = 1, \dots, K$$

$$g_L \leq g(t, y, x, \lambda, p) \leq g_U$$

with respect to

$$y_{0,L} \leq y_0 \leq y_{0,U}$$

$$y_{f,L} \leq y_f \leq y_{f,U}$$

$$x_{0,L} \leq x_0 \leq x_{0,U}$$

$$x_{f,L} \leq x_f \leq x_{f,U}$$

$$t_{0,L} \leq t_0 \leq t_{0,U}$$

$$t_{f,L} \leq t_f \leq t_{f,U}$$

$$y_L \leq y(t) \leq y_U$$

$$x_L \leq x(t) \leq x_U$$

$$p_L \leq p \leq p_U$$

$$\lambda(t)$$

Computation time in MocoInverse is shorter than in the other problems due to the low number of unknowns, therefore it is the simplest problem which can be faced with Moco [12, 13].

2.4.2 MocoTrack

In these types of problem, there exists information about the motion performed by the model, this information normally consists of experimental data captured by sensors. However, kinematics becomes unknown, so it is one of the variables to obtain, along with controls and states. The objective deals with fulfill the equations of motion (and the rest of constraints) while minimizing as much as possible the deviation of the resulting motion from the experimental data [12, 13].

Figure shows the representative scheme of problems of this nature:

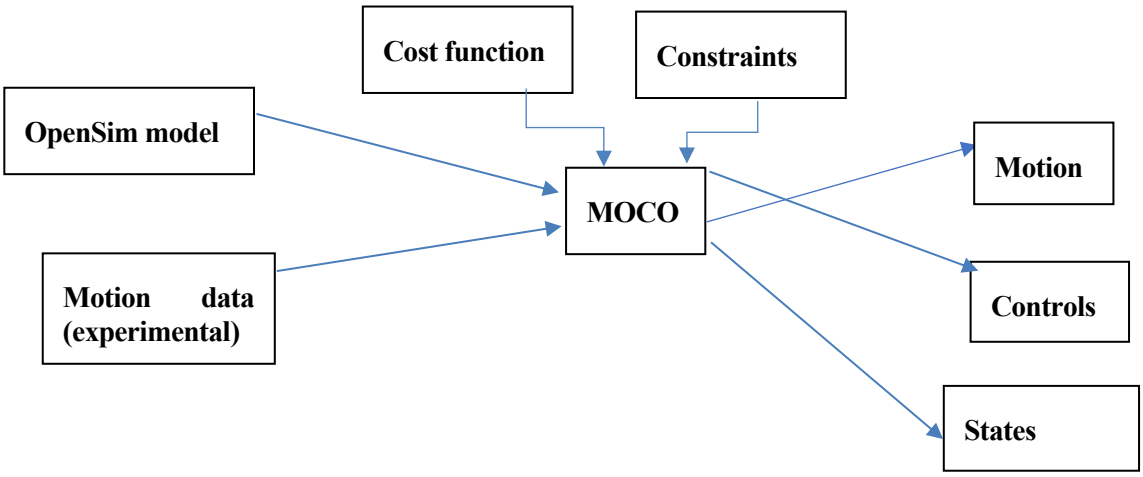


Figure 2.4. Operating scheme in MocoTrack problems

It takes longer to compute in comparison to MocoInverse, as the motion is part of the objective function [12].

2.4.3 MocoStudy

It is the most complex case study approached in Moco, and the most general at the same time, as its formulation is equivalent to the one presented in the sub-chapter dedicated to the optimal control problem. Motion is completely unknown, as there is no data about it, therefore, its computation time is the longest by far [12, 13]. This is the scheme corresponding to MocoStudy:

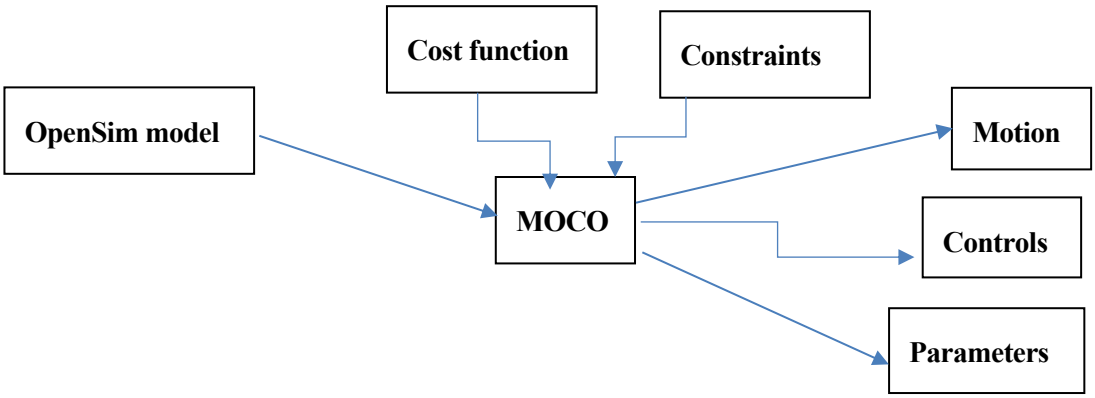


Figure 2.5. Operating scheme of MocoStudy problems.

3 MUSCLE-TENDON MODELLING

Muscles conform a fundamental part in OpenSim models, as they are responsible to exert the needed force to perform motion [16]. They are the actuators in biomechanical problems. For this reason, this chapter will start by explaining the purpose of muscles in models, and the use of Coordinate Actuators as a complementary tool in dynamic problems.

Next, and focusing exclusively on muscles, Hill's biomechanical model will be presented, as it contains the elements needed to represent the muscle. Then mathematical models used to study muscles' behaviour will be presented, including the two differential equations which regulate activation and contraction dynamics, as well as the characteristic curves needed to express the relationships between the diverse parameters required to study muscle motion. These curves vary depending on the model used; in this research, two muscle models will be described: Thelen, 2003 [17, 18] and De Groote & Fregly, 2016 [19].

3.1 Muscles and Coordinate Actuators

It is just commented that muscles are those which generate the required torque to make the musculoskeletal model to move, acting in fact as actuators [20]. However, it can happen that muscles present in the model were not able to produce this torque due to diverse cause: a bad chosen point of application, small maximum force, or simply that the geometry of the system did not allow the muscles to fulfill their mission completely.

This is not a problem in this project, in fact, the mechanism studied here is the four-bar linkage, which in the first phase of the work has no muscles (then, some muscles will be attached). In their place, Coordinate Actuators are used. These are virtual actuators used by the software to breed the required torque when there are no muscles, or they cannot produce force enough to fulfill the equations of motion. In any case, in this work, the power produced will always be the sum of the power provided by the muscles and the Coordinate Actuators:

$$P_{total} = P_{muscles} + P_{Coord.Actuators} \quad \textbf{(3.1)}$$

There is one Coordinate Actuator associated to each coordinate, so that each one could perform its whole torque associated even if muscles could not. It is fundamental to understand the physical meaning of the appearance of Coordinate Actuators in the motion: it means that muscles cannot generate by themselves the required motion, it is more optimal that this force may be provided from an external device (for example, an exoskeleton) than from the human muscles: the importance of exoskeletons comes now into play. Maybe muscles cannot generate the power, or maybe they can, but it results better in terms of the objective function that this power were supplied by external actuators. Exoskeletons are the way to implement these Coordinate Actuators in real models, providing this force that muscles cannot by themselves.

3.2 The 3-element Hill's model

Hill's model [21, 22, 23] serves to represent the muscle dynamics, which involves contraction and activation. This model contains 3 elements which represent the union muscle-tendon that will allow the OpenSim model to move. The Figure 3.1 serves to represent it:

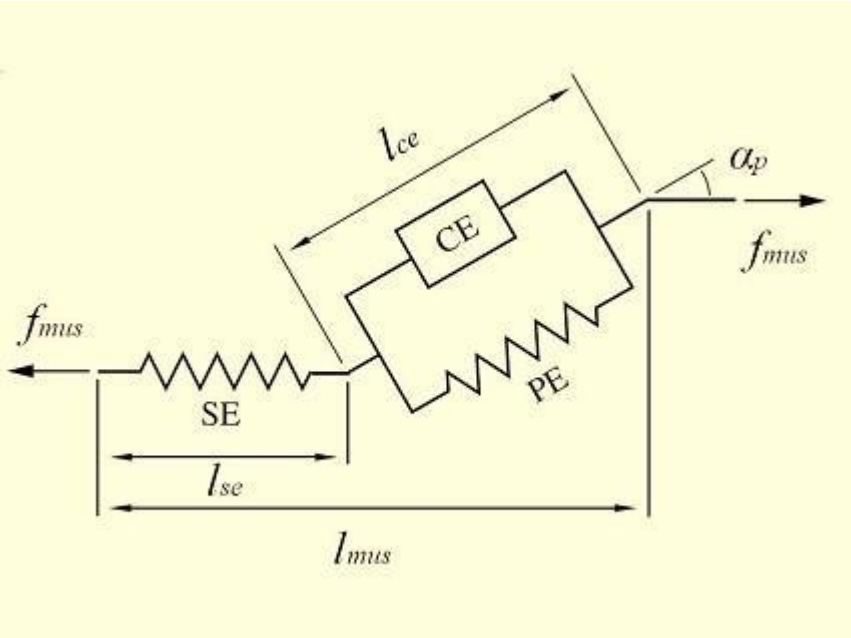


Figure 3.1. 3-element Hill's model. Martínez Reina, J. [21]

In Hill's model, the muscle is represented by the couple CE-PE, where CE is the contractile element, which generates motion due to the interactions between actine and myosine proteins, while PE is the non-linear elastic parallel element, and represents the stiffness due to connective tissue between muscle fibers. The SE (non-linear elastic series element) constitute the stiffness associated to the tendon. For its part, the pennation angle (α_p) is the angle between the longitudinal axis of the generated force, and the direction of muscle fibers. Other important parameters are the tendon length (l_{se}), the contractile element length (l_{ce}) and the muscle length and force (l_{mus} and f_{mus}, respectively) [23].

Some important relationships may be stablished between these parameters, [24, 25]. In fact, the previously defined lengths can be related through the pennation angle following the equation 3.2:

$$l_{mus} = l_{se} + l_{ce} * cos\,(\alpha_p) \qquad \textbf{(3.2)}$$

Also, denoting f_{pe} as the force generated by the parallel element PE, this can be related with f_{mus}, f_{se} and f_{ce} according to the equation 3.3:

$$f_{mus} = f_{se} + (f_{pe} + f_{ce}) * cos\,(\alpha_p) \qquad \textbf{(3.3)}$$

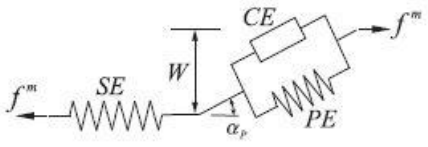


Figure 3.2. Parameters in Hill's muscle model. García Vallejo [25].

We may also define the parameter W, which determines the muscle thickness [25]. It can be defined as the perpendicular component of the CE-PE set to the tendon line of action. A relationship between α_p, l_{ce} and W may be stablished in the equation 3.4:

$$cos\left(\alpha_p\right) = \sqrt{1 - \left(\frac{W}{l_{CE}}\right)^2} \tag{3.4}$$

In this research, pennation angle will be considered constant, so there will always exist a direct relationship between muscle thickness and length. Then, applying the relationship 3.4 to equation 3.2, the expression 3.5 is obtained:

$$l_{mus} = l_{se} + l_{ce} * \sqrt{1 - \left(\frac{W}{l_{CE}}\right)^2} \tag{3.5}$$

Derivating this equation with respect to time, the relationship between velocities is obtained in 3.6:

$$\dot{l_{mus}} = \dot{l_{se}} + \dot{l_{ce}} * \sqrt{1 - \left(\frac{W}{l_{ce}}\right)^2} + l_{ce} * \frac{1}{2} * \left(1 - \left(\frac{W}{l_{ce}}\right)^2\right)^{-\frac{1}{2}} * \left(-2\frac{W}{l_{ce}}\right) * \left(\frac{-W * \dot{l_{ce}}}{l_{ce}^2}\right) \tag{3.6}$$

This expression may be expressed in a simpler way (Equation 3.7)

$$\dot{l_{mus}} = \dot{l_{se}} + \dot{l_{ce}} * \left(\alpha_p\right) + \frac{sin^2(\alpha_p)}{cos\left(\alpha_p\right)}) \tag{3.7}$$

Which becomes, applying trigonometric rules, the equation 3.8, much easier to understand:

$$v_{mus} = v_{se} + \frac{v_{ce}}{cos(\alpha_p)} \tag{3.8}$$

Now, activation and contraction dynamics will be introduced.

3.3 Muscle dynamics

Muscle dynamics, it is said, the study of the causes for muscle motion, is governed by two differential equations (Nagano and Gerritsen, [26]):

$$\dot{a} = (u - a)/(\tau(a, u)) \quad \textbf{(3.9)}$$

$$\dot{f_{mus}} = f(a, l_{mus}, v_{mus}, f_{mus}) \quad \textbf{(3.10)}$$

The first equation corresponds to the relationship between the muscle activation (and its first derivative) and neural excitation, while the second one deals with the dependance of muscle force (and its first derivative) with the activation, the muscle length, and the velocity of contraction.

Parting from the Equation 3.9, the first derivative of the activation appears at the left side, while at the right side we can found the excitation u, the activation a, and a function τ depending on these two parameters. The activation is set to vary between 0 (full absence of contraction) and 1 (total contraction). This magnitude depends on the quantity of motor units recruited and firing frequency [27]. The muscle excitation is also bounded between 0 (null excitation) and 1 (full excitation), and it represents the net effect of motor neuron recruitment and their firing frequency [27].

The function $\tau(a, u)$ depends on the activation and a time constant that scales the difference between the activation and excitation. Its value is dependent on the following conditions:

$$\tau(a, u) = t_{act} * (0.5 + 1.5 * a) \quad si \quad u > a \quad \textbf{(3.11)}$$

$$\tau(a, u) = t_{deact}/(0.5 + 1.5 * a) \quad si \quad u \leq a \quad \textbf{(3.12)}$$

Where t_{act} and t_{deact} are activation and deactivation time constants, respectively, and receive values of 10 ms for activation time and 40 ms for deactivation time. The Figure 3.3 shows the graphical representation of the muscle activation and the neural excitation through time.

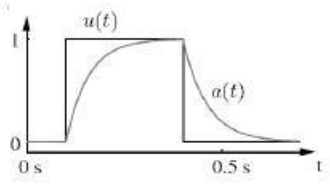


Figure 3.3. Activation and deactivation time constants. [25]

Concerning the second differential equation in muscle dynamics [25], if the expression is integrated at both sides the relation between muscle force and its three main parameters (activation, muscle length and contraction velocity) is obtained:

$$f_{mus} = f(a, l_{mus}, v_{mus}) \quad \textbf{(3.13)}$$

So as to maintain the joint between the muscle fibers and the tendon, the equilibrium equation between the force produced by the tendon and the one exerted by muscle fibers must be fulfilled:

$$f_{max}^{iso} * \left(\widetilde{f_{ce}} + \widetilde{f_{pe}}\right) * cos\left(\alpha_p\right) - f_{max}^{iso} * \widetilde{f_{se}} = 0 \quad \textbf{(3.14)}$$

Being f_{max}^{iso} the maximum isometrical force (defined for each muscle), $\widetilde{f_{se}}$ the normalized tendon force, while $\widetilde{f_{ce}}$ and $\widetilde{f_{pe}}$ are the active and passive forces, respectively, generated in the muscle fibers. All these forces have a dependance with muscle parameters.

When solving an Inverse Dynamics problem in OpenSim Moco, in the results file generated the normalized tendon force appears. As known in equation 3.14, this force is related to the one exerted by fibers. Moreover, there exists a relation between the tendon force and its length:

$$\widetilde{f_{se}} = f(l_{se}) \quad \textbf{(3.15)}$$

This equation will depend on the muscle model in use. Once the tendon length is known, it is simple to obtain the fiber length from the total muscle length, through the equation 3.16:

$$l_{ce} = \frac{l_{mus} - l_{se}}{cos\left(\alpha_p\right)} \quad \textbf{(3.16)}$$

Now the contractile element length has been obtained, the passive force can be calculated due to the relationship between them:

$$f_{pe} = f(l_{ce}) \quad \textbf{(3.17)}$$

This is because passive force appears when muscle fibers are stretched beyond their slack length. Under this threshold, no passive force appears.

Then, active forces may be calculated, for which two ways can be considered. Firstly, they can be obtained from the total muscle force. As it is known from the equilibrium equation 3.14, muscle force is equivalent to tendon force. The normalized tendon force appearing in the results file from MocoInverse must be multiplied by the maximum isometrical force, getting this way the second term in the equilibrium equation 3.14, mentioned before. Knowing the total muscle force and the passive one, it is easy to calculate the active force:

$$f_{ce} = f_{mus} - f_{pe} \quad \textbf{(3.18)}$$

The second way consists of obtaining the active force through its relationships with muscle parameters. In fact, this force can be expressed the following way [22]:

$$f_{ce} = a * f_l(\widetilde{l_{ce}}) * f_v(\widetilde{v_{ce}}) \quad \textbf{(3.19)}$$

Where a is the muscle activation, while f_l and f_v are functions relating normalized force with normalized fiber length and contraction velocity, respectively.

All these relationships between force and muscle parameters are represented in what are called as musculotendon curves [28]:

- Tendon force length curve: parting from the slack position of the tendon, force starts to be generated when this length gets increased. Figure 3.4 shows an example.

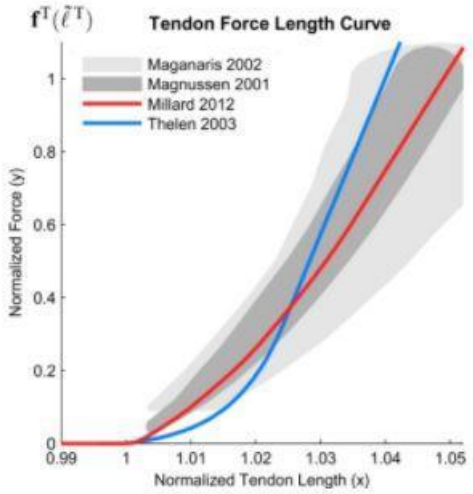


Figure 3.4. Tendon force length curve. OpenSim Documentation [28]

- Passive force length curve: muscle fibers have defined an optimal length that, in case of being surpassed, it starts to generate passive forces. This curve comes represented in Figure 3.5 for diverse muscle models

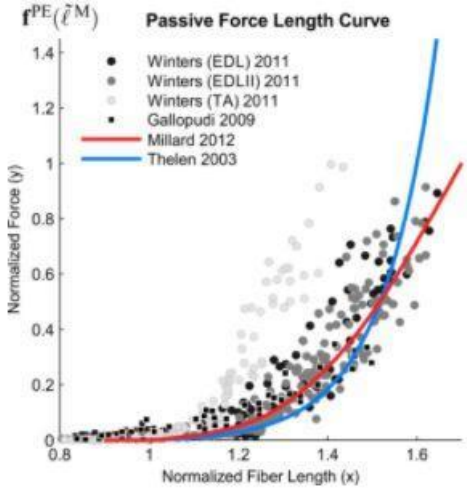


Figure 3.5. Passive force length curve. OpenSim Documentation [28]

- Active force length curve: the optimal fiber length just described is that one which allows the muscle to generate the maximum force possible. If it is surpassed, not only passive forces start to appear, but also active forces get reduced. Of course, under this threshold active forces decrease too. Diverse representations of the curve may be appreciated in Figure 3.6:

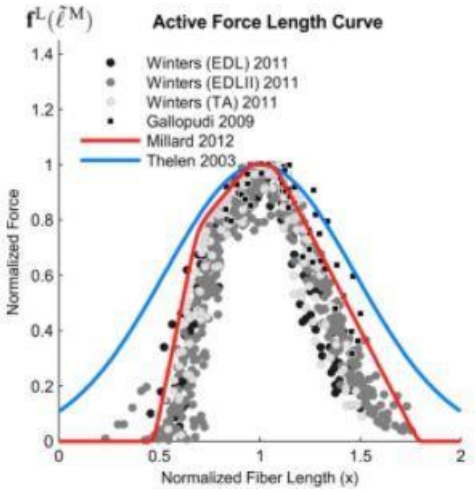


Figure 3.6. Active force length curve. OpenSim Documentation [28]

- Force velocity curve: in the movement, a muscle can be in two different phases: lengthening and shortening. In the first phase, which corresponds to a negative contraction velocity ($v < 0$) the muscle decreases the generated force, while the second phase deals with positive contraction velocity ($v > 0$). The muscles can generate more force than in the previous phase. The Figure 3.7 shows some representations of these curve.

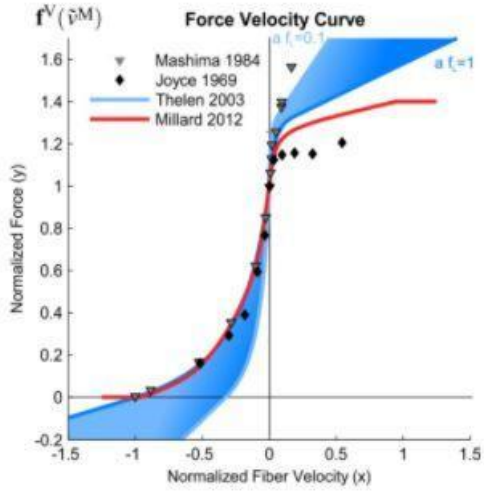


Figure 3.7. Tendon force length curve. OpenSim Documentation [28]

Now that the process to obtain different forces has been explained, some muscle models will be presented.

3.4 Muscle models

3.4.1 Thelen muscle model

This model was developed by Thelen [18]. Starting with the tendon force, in Thelen's model it is necessary to define, at first, tendon strain, ε_t, as the relative displacement of the tendon with respect to its slack length:

$$\varepsilon_t = \frac{l_{se} - l_{slack}}{l_{slack}} \tag{3.20}$$

Now the force exerted by the tendon according to Thelen can be expressed:

$$F_t = F_{max} * 0.10377 * (e^{91*\varepsilon_t} - 1) \qquad if\ 0 < \varepsilon_t < 0.01516 \tag{3.21}$$

$$F_t = F_{max} * (37.526 * \varepsilon_t - 0.26029) \quad if\ \varepsilon_t \geq 0.01516 \tag{3.22}$$

The Figure 3.8 shows the graphic representation of this curve. The exponential stretch is appreciated from the beginning until the strain reaches a value of ε_t= 0.01516, where the lineal part starts.

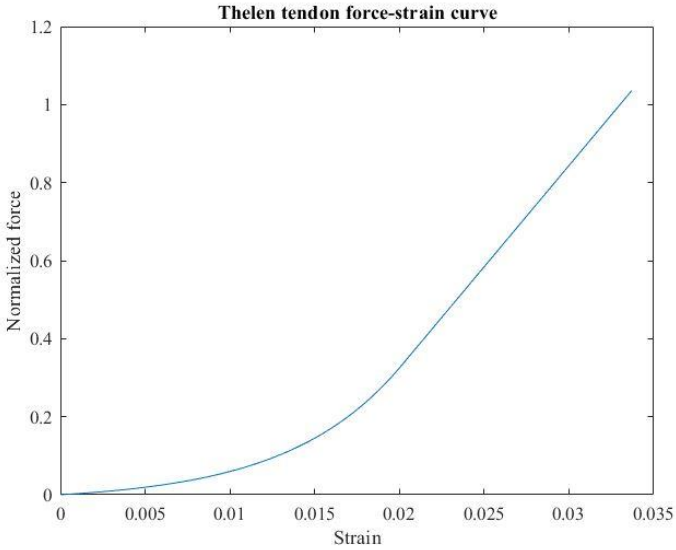


Figure 3.8. Tendon force-strain curve by Thelen. Obtained in Matlab.

With respect to the parallel element, which produces passive forces, the expression depends on the normalized

fiber length, obtained as follows:

$$\widetilde{l_{pe}} = \frac{l_{pe}}{l_{opt}} \tag{3.23}$$

Being l_{opt} the optimal fiber length. This way, the equation to obtain the normalized passive force is the following one:

$$\widetilde{f_{pe}} = 1 + \frac{k_{pe}}{\varepsilon_{ce}} * \left(\tilde{l}_{ce} - (1 + \varepsilon_{ce})\right) \quad if \ \tilde{l}_{ce} > \ 1 + \varepsilon_{ce} \tag{3.24}$$

$$\widetilde{f_{pe}} = \frac{e^{\frac{k_p * (\widetilde{l_{ce}} - 1)}{\varepsilon_{ce}}}}{e^{k_{pe}}} \quad if \ \tilde{l}_{ce} \leq \ 1 + \varepsilon_{ce} \tag{3.25}$$

Where k_{pe} is a shape factor got experimentally, and ε_{ce} is the passive muscle strain at the maximum isometrical force (in this work, k_{pe} = 4 and ε_{ce} = 0.6). The graphic representation comes as follows in the Figure 3.9:

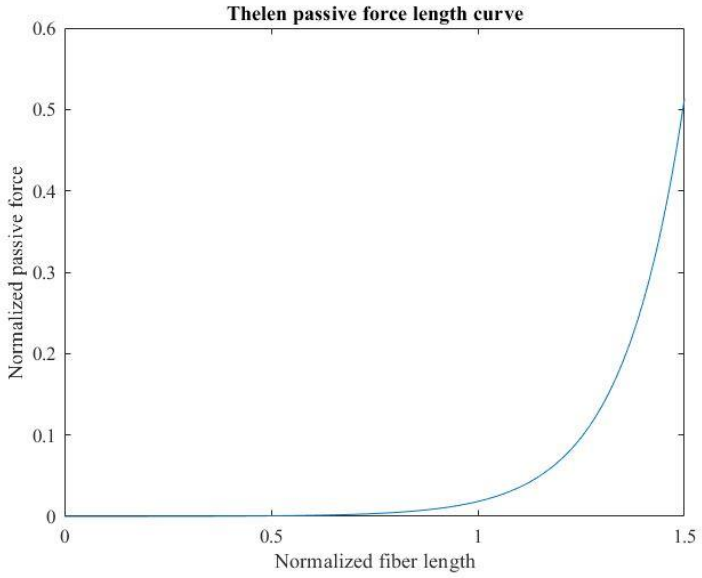


Figure 3.9. Tendon force-strain curve by Thelen. Obtained in Matlab.

The next step would be to calculate the effect of the fiber length in the active force. So that, the next exponential expression is used:

$$f_L = e^{\frac{-(\tilde{l}_{ce}-1)^2}{\gamma}} \tag{3.26}$$

Where γ corresponds to the half width of the curve $\frac{1}{e}$. In this project, a value of 0.45 has been assigned to this parameter. The Figure 3.10 shows the graphic relation between active force length parameter and the normalized fiber length:

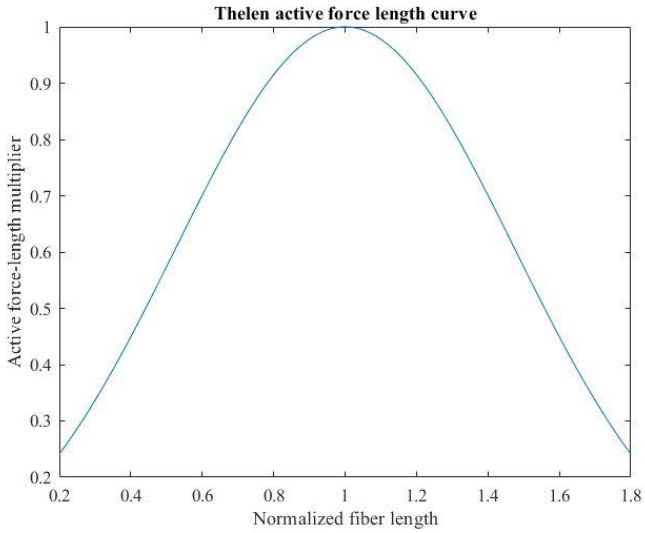


Figure 3.10. Active force-length curve by Thelen. Obtained in Matlab

Last, the relationship force-velocity must be represented. This expression is written in function of the normalized contraction velocity, which comes defined as follows:

$$\widetilde{v_{ce}} = \frac{v_{ce}}{v_{max}} \tag{3.27}$$

Resulting v_{max} the maximum contraction velocity of the muscle, in this work it is assigned a 10 m/s value. Now the expression force-velocity can be written:

$$f_v = \frac{1+\tilde{v}_{ce}}{1-\frac{\tilde{v}_{ce}}{A_f}} \quad if\ \tilde{v}_{ce} \leq 0 \tag{3.28}$$

$$f_v = \frac{1+\tilde{v}_{ce} * f_{v_{max}} * (2+\frac{2}{A_f})}{1+\tilde{v}_{ce} * \frac{(2+\frac{2}{A_f})}{f_{v_{max}}-1}} \quad if\ \tilde{v}_{ce} > 0 \tag{3.29}$$

Where $f_{v_{max}}$ is the maximum normalized lengthening force (its assigned value is 1.8), and A_f a shape factor (its value has been set to 0.25). The curve representing these equations is displayed in Figure 3.11:

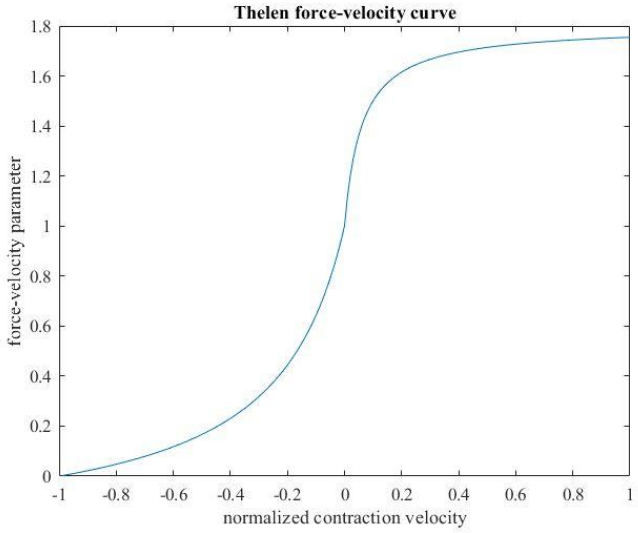


Figure 3.11. Force velocity curve by Thelen. Obtained in Matlab

3.4.2 De Groote-Fregly muscle model

This model was developed by De Groote, Fregly et al. [19]. Unlike Thelen's model, which defined a tendon strain, De Groote-Fregly's considers a normalized tendon length as follows:

$$\tilde{l}_{se} = \frac{l_{se}}{l_{slack}} \tag{3.29}$$

Being l_{slack} the tendon slack length. This adimensional magnitude has a lower bound of zero, as the tendon cannot have a shorter length than its slack length.

Known this, the equation relating the tendon force and its length is written the following way:

$$\tilde{l}_{se} = \frac{ln\,(5*\tilde{f}_{se}+0.25)}{48} + 0.995 \tag{3.30}$$

Its graphic representation comes displayed as follows in Figure 3.12:

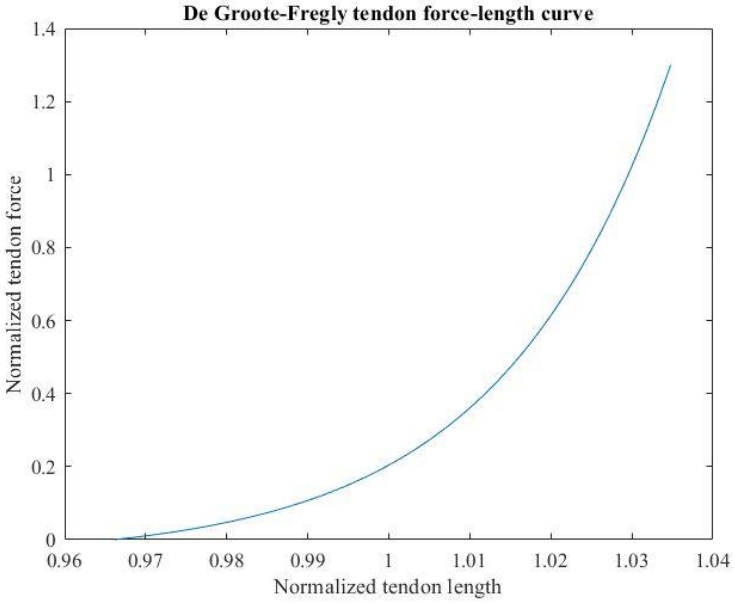


Figure 3.12. Tendon force-length curve by De Groote-Fregly. Obtained in Matlab

Concerning the passive force-length relationship, De Groote-Fregly's model utilizes the same expression as Thelen's, but without considering the lineal stretch, the whole curve corresponds to the exponential expression:

$$\widetilde{f_{pe}} = \frac{e^{\frac{k_p * (\widetilde{l_{ce}} - 1)}{\varepsilon_{ce}}}}{e^{k_{pe}}} \qquad \textbf{(3.31)}$$

Where, as already explained in Thelen's model, k_p corresponds to an experimental shape factor (a value of 4 has been stablished for it), while ε_{ce} is the passive muscle strain at maximum isometric muscle force (it has been set to 0.6). The curve corresponding to the equation is shown in Figure 3.13:

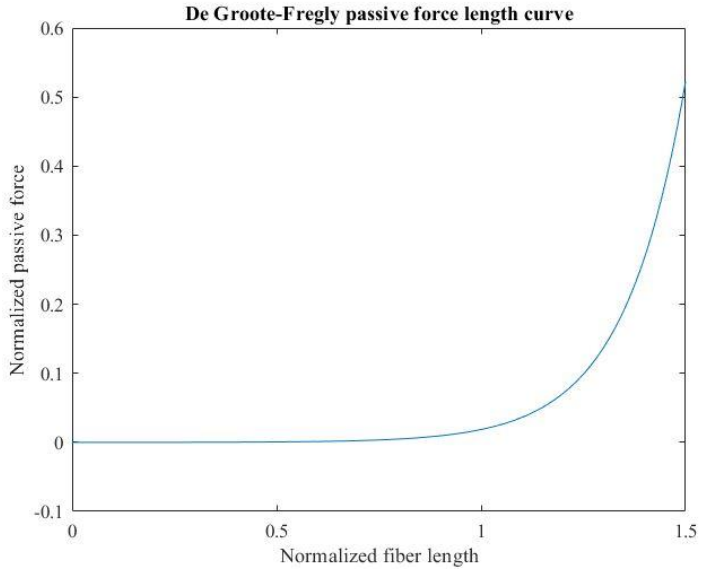


Figure 3.13. Passive force-length curve by De Groote-Fregly. Obtained in Matlab.

With regards to the active force length characteristic curve, De Groote-Fregly's model sets it as the sum of three Gaussian expressions, which contain several experimental values. These relationships are written as follows:

$$f_l = \sum_{i=1}^{3} b_{1i} * e^{\left(\frac{-0.5*(\widetilde{l_{ce}}-b_{2i})^2}{b_{3i}+b_{4i}*\widetilde{l_{ce}}}\right)} \quad \textbf{(3.32)}$$

Being b_{ij}, i=1…4, j=1…3 experimental parameters whose value comes collected in the Table 3.1:

Table 3.1. Experimental parameters in passive force-length equation by De Groote-Fregly [19]

b_{11}	b_{21}	b_{31}	b_{41}	b_{12}	b_{22}	b_{32}	b_{42}	b_{13}	b_{23}	b_{33}	b_{43}
0.815	1.055	0.162	0.063	0.433	0.717	-0.03	0.02	0.1	1	0.354	0

Its graphic representation comes displayed in the Figure 3.14:

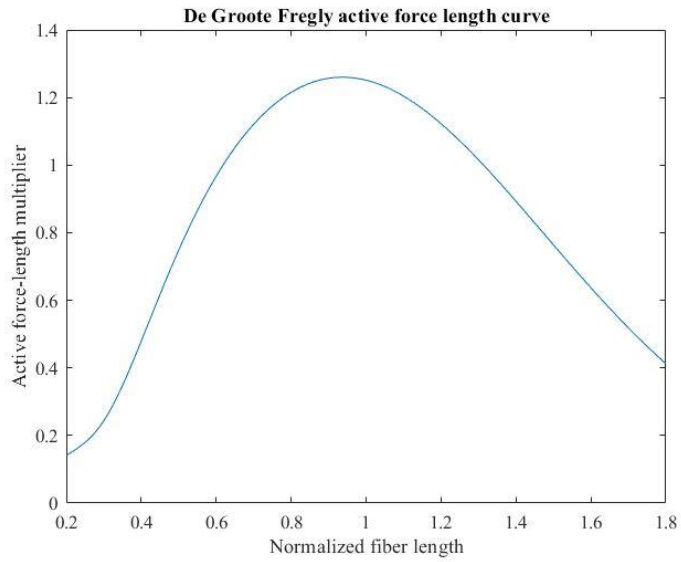


Figure 3.14. Active force-length curve by De Groote-Fregly. Obtained in Matlab

With respect to the force velocity curve, it is described by a logarithmic function which contains some predefined parameters:

$$f_v = d_1 * ln\left[(d_2 * \widetilde{v_{ce}} + d_3) + \sqrt{((d_2 * \widetilde{v_{ce}} + d_3)^2 + 1)}\right] + d_4 \quad \textbf{(3.33)}$$

Where the parameters d_i , i = 1…4 have been assigned the next values:

Table 3.2. Parameters in force-velocity equation by De Groote-Fregly [19]

d_1	d_2	d_3	d_4
-0.3183	-8.1492	-0.3741	0.8856

The force-velocity curve according to De Groote-Fregly's model results to be the following one:

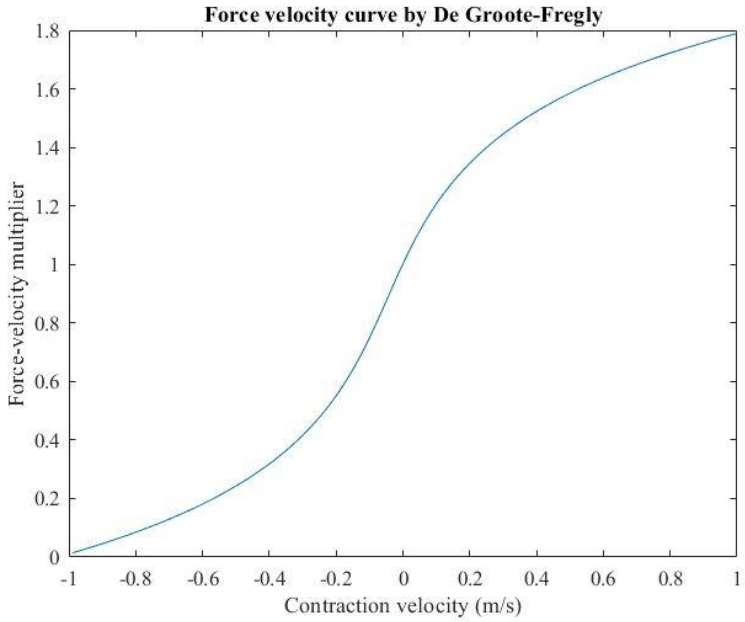


Figure 3.15. Force-velocity curve by De Groote-Fregly. Obtained in Matlab.

4 THE CLOSED-LOOP PROBLEM

As already explained, the intention of this work is to study the possibility of using OpenSim Moco to study dynamics in human people wearing exoskeletons. When attached to a person, the union formed by the individual and the exoskeleton constitutes a muscled closed-loop mechanism. This way, the objective of the project is summed up in checking if OpenSim Moco can solve Inverse Dynamics problems, firstly, in closed-loop mechanisms, and then, in muscled closed-loop mechanisms.

This chapter explains, in a theoretical and concise manner, how the objective right just described is faced in this work. It starts with the way OpenSim approaches Inverse Dynamics, as well as its form to solve open-loop and closed-loop mechanisms, and the use of Moco to face these problems. The four-bar linkage, the closed-chain mechanism used in this project, will be presented. By last, the analytical inverse dynamics problem applied to a multibody system will be introduced, as this is the problem whose result will be compared to the Inverse Dynamics made in Moco.

4.1 Open loop mechanisms in OpenSim

As introduced earlier, OpenSim allows to develop musculoskeletal models and dynamic simulations to study its movement. Equations of motion are formulated by OpenSim by using Simbody, an open-source multibody dynamics solver [20]. OpenSim models are constituted by bodies, joints and, if needed, constraints:

a) Bodies: they are rigid solids connected by joints, and conform the main blocks in the model. Generally, each body has assigned a joint which links it to a previous body (this body is considered a parent one), and it is also connected to a successive body (a child one) by another joint [20].

b) Joints: they serve to stablish the unions between bodies. Models are built following a chain of bodies linked by joints, where each joint relates two bodies: the parent and the child one, except for the initial body (normally the ground, it is always a parent one) and the final bodies (there are as many as different chains the model contains; final bodies are always child ones).[20]

   Joints have a great importance as they define the coordinates of the model, it is said, the parameters which will determine the motion of the model. In building the joint, not only the coordinates must be declared, but also the position and orientation with respect to the parent and child bodies. There are different available joints in OpenSim according to the number of coordinates they introduce, here the most used in this Project will be described:

   - Ball joints: they introduce 3 rotational coordinates between bodies, blocking any possible translation between them.
   - Pin joints: they only introduce one rotational coordinate, through the Z-axis common to both bodies, restricting the other two restrictions and the three translations.
   - Weld joints: they fuse the two bodies, avoiding any possible movement, rotational or translational. No coordinates are defined in Weld joints.

- Free joints: they allow every possible movement between bodies: 3 rotations and 3 translations. They are not in fact a joint, but OpenSim creates them by default if there is a body without joints to another body.

- Custom joints: they are completely defined by user, included the number of coordinates and their type. They are useful to describe more complex joints, frequent in human body (e.g. the scapulothoracic joint, containing 4 rotational coordinates).

  In custom joints, the user must also define the Spatial Transform, it is said, the relationship between the generalized coordinates introduced by the joint, and the translations and orientations with respect to the X-Y-Z axes.

c) Constraints: they are needed in OpenSim to close the chains formed by bodies and their joints, because a chain cannot be closed by a joint. The chain must always start with an initial parent body and end with a finish child body [20].

   In OpenSim constraints differ from joints on the fact that they do not introduce coordinates, they just restrict coordinates previously defined by joints. There are three types of constraints available in OpenSim:

   - Point constraints: they force bodies to be linked through a point, blocking any possible translation between them.

   - Weld constraints: they completely fuse the bodies, restricting any possible movement, translation or rotation.

   - Coordinate Coupler constraints: these express a dependent coordinate in function of an independent coordinate.

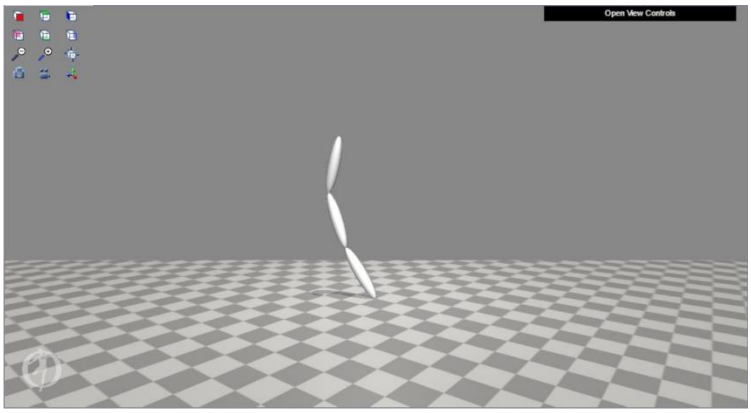


Figure 4.1. Example of open-loop mechanism in OpenSim GUI: 3-bar pendulum. [6]

These models typically represent parts from the human model, or even an entire individual itself, so they are frequently open-chained mechanisms, this is, these models have always an initial parent body (it uses to be the ground) and one or more ending child bodies.

It is possible to perform an Inverse dynamics analysis in OpenSim. Firstly, it is necessary to get a prescribed motion, what is obtained in an Inverse kinematics analysis (paragraph 4.2), and then, along with data concerning reaction forces and mass and inertia properties, Inverse dynamics (paragraph 4.3) can be done [17]. Results will appear in OpenSim GUI (Graphical User Interface).

4.2 Inverse kinematics in OpenSim

Inverse kinematics is one of the most useful tools which can be found in OpenSim, as we can use it to study the movement of the model. Through this tool, and parting from the experimental data, the software calculates for each time frame values for each generalized coordinate in the model, trying to match as accurate as possible the experimental data (markers and unprescribed coordinates), by minimizing the following cost function [29]:

$$f = \sum_{i \in markers} w_i \left\| x_i^{exp} - x_i(q) \right\|^2 + \sum_{j \in unprescribed\ coords} w_j (q_j^{exp} - q_j)^2 \qquad \textbf{(4.1)}$$

This cost function consists of the sum of two terms representing weighted squared errors of markers and unprescribed coordinates, respectively, where:

- **q** is the vector containing generalized coordinates for each time frame.
- **x (q)** is the vector containing the position of virtual markers, which depend on generalized coordinates.
- $\mathbf{x}^{exp}$ is the vector of positions of experimental markers.
- w is the weight associated to each marker; this is assigned in the XML model file. This term multiplies the squared error between virtual and experimental markers.
- q^{exp} represents the unprescribed experimental coordinates.
- ω_j is the weight associated to the unprescribed coordinates.

In this project, no prescribed coordinated have been used in inverse kinematics.

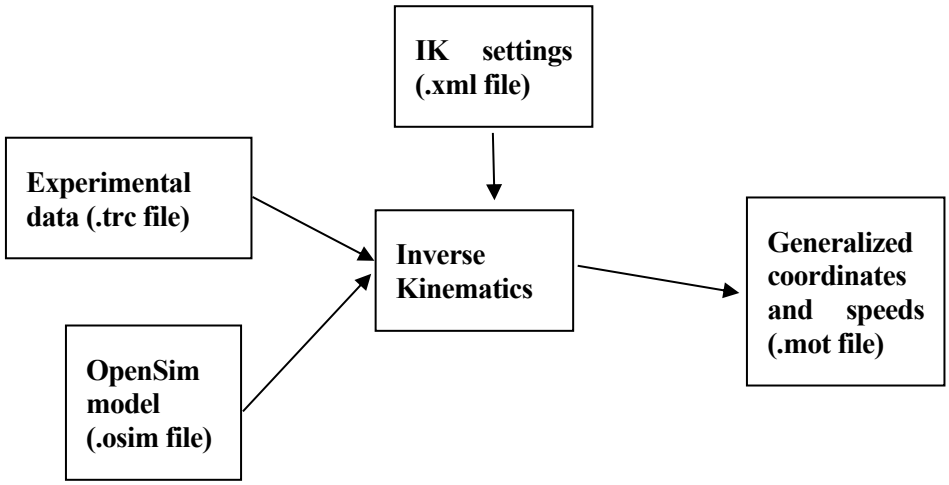


Figure 4.2. Operating scheme of Inverse Kinematics in OpenSim

The scheme in Figure 4.2 shows how Inverse Kinematics works in OpenSim. It parts from the model, (coded in an .osim file) which contains all the information about bodies, joints and constraints between them, and the experimental data obtained via sensors located in determined positions in the model, collected in a .trc file [29]. Additionally, OpenSim requires some settings (initial and final time instants of the simulation, weights associated to coordinates, etc.), which are included in an .xml file.

OpenSim uses all these data to solve in Inverse Kinematics problem by calculating the generalized coordinates and velocities, minimizing the objective function (Equation 4.1).

4.3 Inverse dynamics in OpenSim

Inverse Dynamics is another important tool present in OpenSim whose utility is to calculate the generalized forces responsible for a given movement [30]. For that, this tool uses the Newton-Euler equations of motion to obtain these torques and forces for each generalized coordinate:

$$\boldsymbol{M}(\boldsymbol{q}) * \ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \qquad \textbf{(4.2)}$$

Where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the generalized coordinates, velocities, and accelerations, $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ the vector of Coriolis and centrifugal forces, $\boldsymbol{G}$ the vector of gravitational forces and $\boldsymbol{\tau}$ the vector of generalized forces, being this last one the only unknown in the problem. As mentioned in paragraph 4.1, these equations are formulated by OpenSim by using Simbody.

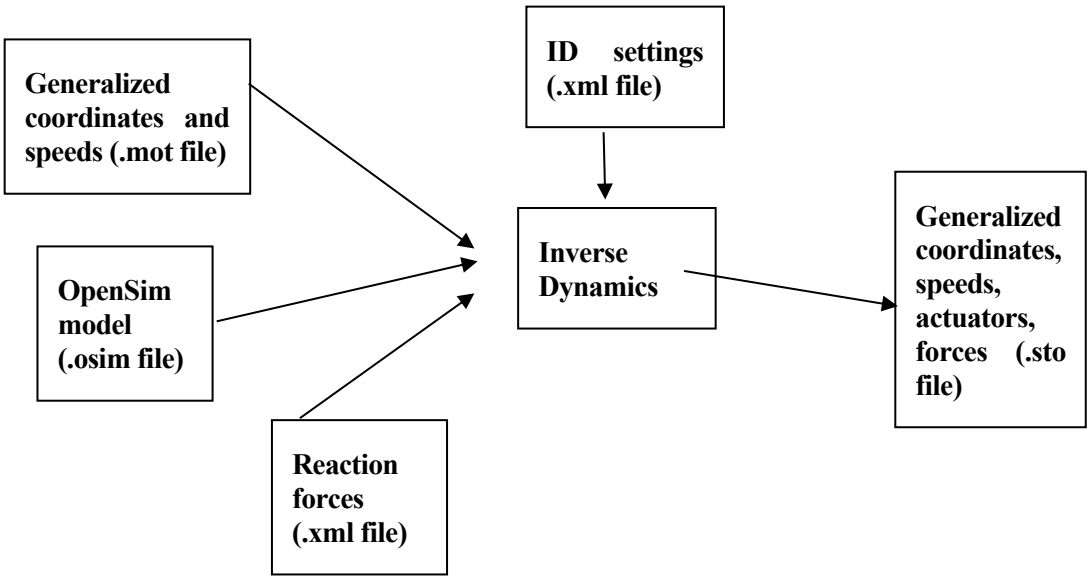


Figure 4.3. Operating scheme of Inverse Dynamics in OpenSim.

The Figure above represents how Inverse Dynamics works in OpenSim. It parts from the results obtained in the

kinematic analysis (positions and velocities), and adds information about external reaction forces, as they are required to solve the problem. Also, the .osim model is needed as it contains all information concerning mass and inertia properties [30].

4.4 Closed-loop mechanisms

As indicated in the introduction of this chapter, the object of this work is to check the feasibility to use OpenSim Moco to solve dynamics in exoskeletons worn by people. The link between the human body and the exoskeleton cannot be represented by an open-loop mechanism, as the exoskeleton must be connected to the body through two points (see Figure 4.4)

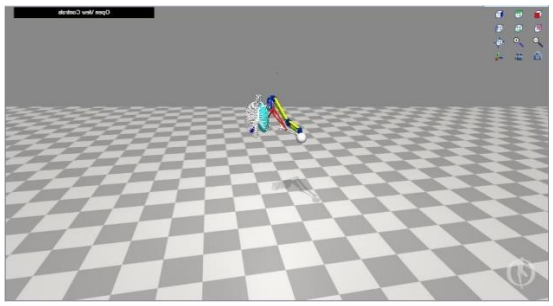


Figure 4.4. Example of closed-loop model in OpenSim [31]

As indicated in paragraph 4.1, chain containing bodies cannot be closed by using joints, because considering the hierarchical structure of bodies in an OpenSim model, starting with the initial parent body, and finishing with the final body, a body cannot be parent to another which is in a previous place in this hierarchy [17].

Therefore, the only mode to close the chain is to use constraints. A constraint can be created between the final and the initial body in the chain. This restriction will limit the motion between them, working practically as another joint. Other form to close the mechanism is by creating an additional body, identical to the first one in the chain which is desired to be closed and fusing it with this identical body through a weld constraint (or a double point constraint), so that these bodies have no difference between them. The chain can be closed by using a joint between the last body and this additional body just created.

The simplest closed-loop mechanism which may be built is the four-bar linkage, whose motion can be modeled with an only degree of freedom. Despite its simplicity, its results can be extrapolated to the rest of closed-loop models, so it has been the chosen model in this project to study its Inverse Dynamics. Also, the muscled version of the 4-bar will serve to extrapolate its results to a human-exoskeleton model.

4.5 Use of Moco to solve Inverse Dynamics in closed-chain mechanisms

OpenSim can perform Inverse Dynamics in open-loop mechanisms, as it is one of its main functionalities, but this is not possible in closed-loop models, because the results obtained are incorrect. This will be verificated in Chapter 6, when solutions from the analytical 4-bar Inverse Dynamics problem and its analogous in OpenSim

are compared in terms of equivalent dynamical power.

OpenSim Moco solves this circumstance. Chapters 5 and 6 will demonstrate that Moco can be used to obtain reliable Inverse Dynamics' solutions, as the dynamical power obtained in the MocoInverse problem results to be the same as in the analytical problem.

4.6 Analytical Inverse Kinematics and Dynamics

The last paragraph in this Chapter will serve to introduce the analytical general formulations of Inverse Kinematics and Inverse Dynamics, valid for any multibody system [32, 33]. The results obtained by implementing these problems will be compared with those from MocoInverse.

4.6.1 Kinematic problem

Every multibody system contains a set of constraints that relate the different coordinates. This set of equations can be expressed this way:

$$\boldsymbol{C(q) = 0} \qquad \textbf{(4.3)}$$

Where **q** is the vector containing every generalized coordinates and **C** is the vector containing every constraint. There are two types of constraints: geometrical, which relate coordinates between them, and kinematical, which relate coordinates with time. As there are as many equations (constraints) as unknowns (generalized coordinates), the problem has a unique solution and coordinates may be obtained.

If the expression above is derivated with respect to time, generalized velocities can be expressed in function of the generalized coordinates:

$$\boldsymbol{C_q(q) * \dot{q} + C_t = 0} \qquad \textbf{(4.4)}$$

And clearing the unknowns:

$$\boldsymbol{\dot{q} = -{C_q}^{-1}(q) C_t} \qquad \textbf{(4.5)}$$

Where $\boldsymbol{\dot{q}}$ is the vector of generalized velocities, $\boldsymbol{C_q(q)}$ is the jacobian matrix of restrictions and $\boldsymbol{C_t}$ is the first derivative of the constraints matrix with respect to time. The jacobian matrix contains, for each constraint, its first derivative with respect to each coordinate. As coordinates have already been calculated, velocities can also be obtained.

Now, derivating the expression above again with respect to time, generalized accelerations will be got:

$$\boldsymbol{C_q(q) * \ddot{q} + \dot{C}_q(q) * \dot{q} + \dot{C}_t = 0} \quad \textbf{(4.6)}$$

Clearing the accelerations:

$$\boldsymbol{\ddot{q} = -C_q(q)^{-1}(\dot{C}_q(q) * \dot{q} + \dot{C}_t)} \quad \textbf{(4.7)}$$

Being $\boldsymbol{\dot{C}_q(q)}$ the first derivative with respect to time of the jacobian matrix, and $\boldsymbol{\dot{C}_t}$ the second derivative with respect to time of the kinematic constraints.

Now that positions, velocities, and accelerations are known, the kinematic problem have been solved.

4.6.2 Dynamic problem

Having solved kinematics, the Inverse Dynamics problem can be approached. In this manner, the Newton-Euler's system of equations to calculate generalized force and moments is defined [32, 33]:

$$\boldsymbol{M * \ddot{q} + C_q^T * \lambda = Q_{apl} + Q_v} \quad \textbf{(4.8)}$$

Where:

- $\boldsymbol{M}$ is the mass matrix, and given that the coordinates system chosen is centered in the center of gravity of the system, it results to be diagonal, as it contains each body's mass matrix which, thus, are also diagonal:

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M_i} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{M_N} \end{pmatrix} \quad \boldsymbol{M_i} = \begin{pmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & Ig_i \end{pmatrix}$$

  Where N is the total number of bodies involved in the mechanism and m_i and Ig_i correspond, respectively, to the mass and the moment of inertia of the body i. It may be noted that the problem is bidimensional, what makes the individual mass matrix for each body to have only three values: two corresponding to the two planar traslations along X-and-Y-axis; and one corresponding to the rotation around Z-axis

- $\boldsymbol{\ddot{q}}$ is the vector containing every generalized acceleration.
- $\boldsymbol{C_q^T}$ is the transpose jacobian matrix (it will be later defined).
- $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers, each one associated to one constraint.
- $\boldsymbol{Q_{apl}}$ consists on the vector containing external applied forces (not including gravitational forces) in the generalized coordinates.
- $\boldsymbol{Q_v}$ is the vector containing the gravitational generalized forces.

As the accelerations are already known, Lagrange multipliers can be easily achieved:

$$\boldsymbol{\lambda = C_q^{T^{-1}}(Q_{apl} + Q_v - M * \ddot{q})} \quad \textbf{(4.9)}$$

Now, generalized forces can be calculated:

$$\boldsymbol{Q_{gen} = -C_q^T * \lambda} \quad \textbf{(4.10)}$$

Ending with the Inverse Dynamics problem. These generalized forces represent the forces required to apply in each generalized coordinate to fulfill each restriction; in fact, there is a generalized coordinate associated to each generalized force, and one generalized force associated to each restriction. In the next chapter, the whole theoretical set just presented will be particularized for the 4-bar linkage.

5 INVERSE DYNAMICS IN THE 4-BAR LINKAGE

In this chapter, the 4-bar mechanism object of study in this project will be presented, including the description of each bar and the generalized coordinates in the problem, as well as the subsequent modifications made later (the inclusion of two muscles to study the interaction of the muscles with the bars of the mechanism) and the motion subject of study the mechanism is doing.

Then, the Inverse Dynamics problem particularized for the 4-bar mechanism will be introduced and slightly detailed, for both the analytical and the MocoInverse problem, presenting all variables, equations and formulations needed to solve it. In Chapter 7, results from all these problems will be displayed.

5.1 The 4-bar linkage

5.1.1 Introduction and composition

The four-bar linkage is the simplest closed chain mechanism which can be built. This mechanism has been used to prove the validity of the results in a closed chain mechanism, in this case, the simplest possible, as it only has one degree of freedom.

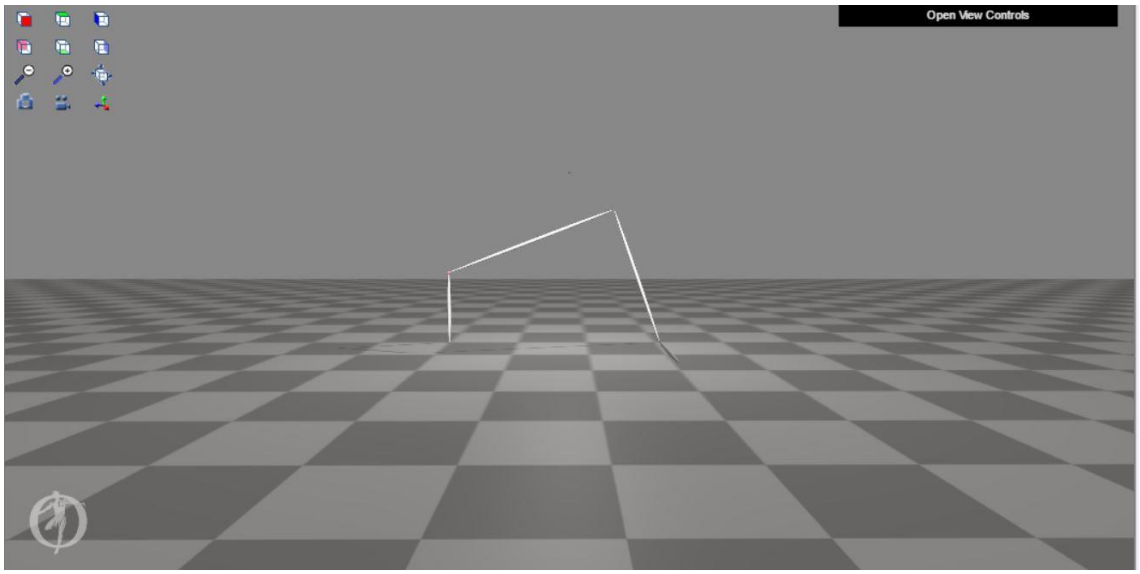


Figure 5.1. 4-bar linkage in OpenSim. [6]

Figure 5.1 shows the four bar linkage used in this project. Different bars composing the mechanism are the following ones:

a) Fixed bar: it does not move as it matches the ground. In this case, it is 1.2 meters long.
b) Input bar: it is linked to the fixed bar through a ball joint (later the reason not to be a pin joint will be explained), and it performs a 360 degrees spin through the full motion cycle. Its length is 0.4 m.
c) Coupler bar: it is connected to the input bar through a pin joint, and transmits the motion to the output bar. It is 1 m long.

d) Output bar: it receives the motion from the coupler bar, whereby it is connected through a pin joint. It is 0.8 m long.

In the literature [32], the output bar is connected to the ground by a pin joint, but as previously explained, in OpenSim a chain cannot be closed only by implementing joints, constraints must be used on their behalf. The proceed followed to close the mechanism here consisted of create a new bar (the one laid in the ground in Figure), and link it to the output bar through a pin joint, and to the fixed bar by a weld constraint. This is equivalent to join the mechanism to the ground with a pin joint.

5.1.2 Grashof condition

In order to perform a complete 360 degrees revolution, the four-bar mechanism must fulfill the Grashof condition, which states that the sum of the lengths of the shortest and the longest bar must be less or equal than the sum of the lengths of the other two bars [32]. This come expressed in the Equation

$$L_1 + L_4 \leq L_2 + L_3 \quad \textbf{(5.1)}$$

Being L_1 and L_4 the longest and shortest bars respectively, and L_2 and L_3 the other two. In this case, L_1 = 1.2 m, L_2 = 1 m, L_3 = 0.8 m and L_4 = 0.4 m, so the Grashof condition gets fulfilled:

$$1.6 \leq 1.8$$

5.1.3 Degrees of freedom

The four-bar mechanism is a bidimensional device so, in order to calculate the number of degrees of freedom (DOFs) the next equation must be used [32]:

$$Nº\ DOF's = 3 * (N - 1) - 2 * P_{II} - P_I \quad \textbf{(5.2)}$$

Being N the number of bodies composing the mechanism (including the fix bar), P_I the number of class-1-restrictions (those which remove one degree of freedom, for example, rotational pairs) and P_{II} the number of class-2-restrictions (those which cancel two degrees of freedom, for example, prismatic pairs). It is direct to know that the 4-bar mechanism contains one DOF:

$$Nº\ DOF's = 3 * (4 - 1) - 2 * 4 = 1$$

But in OpenSim, every mechanism or structure is defined in the tridimensional space, seven plane mechanisms as the 4-bar is considered to have 6 coordinates (3 translations and 3 rotations) instead of only 3 (2 translations and one rotation). This way, the equation to obtain the number of DOFs is the following one [32]:

$$Nº\ DOF's = 6 * (N - 1) - 5 * P_V - 4 * P_{IV} - 3 * P_{III} - 2 * P_{II} - P_I \qquad \textbf{(5.3)}$$

Being P_I, P_{II}, P_{III}, P_{IV} and P_V those kinematic pairs which remove 1, 2, 3, 4 and 5 coordinates, respectively. If the number of DOFs is calculated for a 4-bar mechanism in the 3D space, this result is obtained:

$$Nº\ DOF's = 6 * (4 - 1) - 5 * 4 = -2$$

This negative result has no physical sense; it appears because there are redundant constraints in the mechanism, as OpenSim, as just explained, always considers the model to be tridimensional, assigning thus 6 coordinates to each body. This circumstance makes the model susceptible to lose its capacity to move due to numerical errors from software. That is the reason why it is needed to change any of the introduced pin joints for a ball joint, so as to cancel these redundant restrictions. In this project, the ball joint has been placed in the joint linking the input bar to the fix bar.

5.1.4 Generalized coordinates

Given all introduced joints, finally 6 generalized coordinates define the movement of the 4-bar mechanism:

- $\Theta_{12x,}$ Θ_{12y} and Θ_{12z}, corresponding to the three relative spins between the input bar and the fix bar. In practice, as the motion is plane, Θ_{12x} and Θ_{12y} result to be negligible.

- Θ_{23z}, it is said, the relative rotation between the input bar and the coupler bar.

- Θ_{34z}, the relative spin between the coupler bar and the output bar.

- Θ_{45z}, the relative rotation between the coupler bar and the fix bar. It is also negligible, so the real number of generalized coordinates turns out to be 3.

5.2 Description of the motion

As the 4-bar mechanism has only one degree of freedom, it is enough to use one generalized coordinate to express the movement of the model. In this case, as usual in problems involving this linkage, the coordinate related to the input angle, $\Theta_{12z,}$ (hereafter, $\Theta_{12,}$ as Θ_{12x} and Θ_{12y} are negligible) has been the chosen one.

A 360 degrees spin has been imposed to this coordinate, parting from the following initial value:

$$\theta_2(0) = \Pi\ rad$$

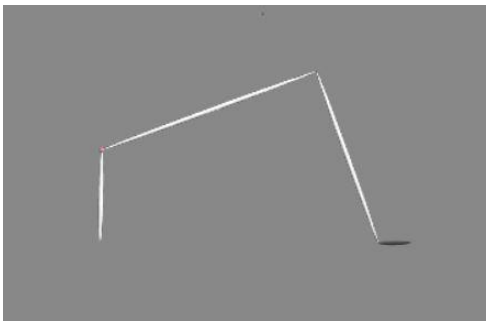
According to the definition of θ_2 given in Figure below, and corresponding to a vertical orientation of the input bar, which moreover spins, during a time span of 6 seconds, at constant velocity:

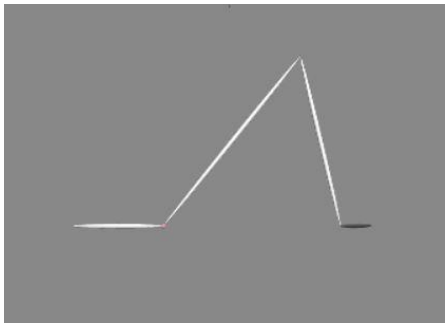
$$\omega_2 = -1.0472\ rad/s$$

The inclusion of the negative sign means that the positive spin is counterclockwise, while the bar is spinning clockwise. This way, two phases can be distinguished in the motion:

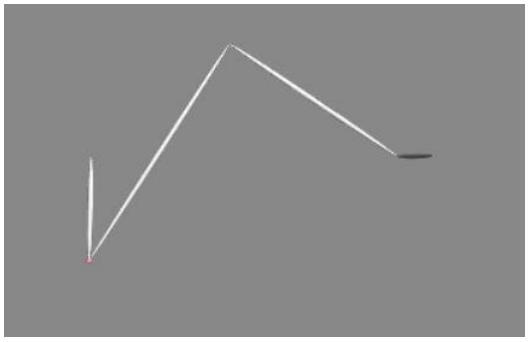
a) A first one, starting with the input bar in vertical position, with the initial condition right presented (it can be seen in Figure 5.2) where the bar is descending during its spin, aligning itself with the ground at the time instant t = 1.5 s (see Figure 5.3) until arriving at a new vertical position at t = 3 s (Figure 5.4).

b) During the second phase of the motion, the bar performs the opposite motion, ascending in its rotation, aligning itself with ground at t =4.5 s (Figure 5.5) and turning to its initial position at t = 6 s (Figure 5.6).





Figures 5.2 and 5.3. The 4-bar at t = 0 and at t = 1.5 s



Figures 5.4 and 5.5. The 4-bar at t = 3 and at t = 4.5 s

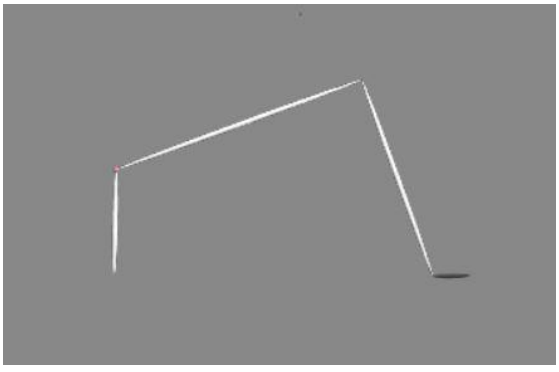


Figure 5.6. The 4-bar at t = 6 s

5.3 Analytical inverse dynamics problem in the 4-bar mechanism

The four bars mechanism is one of the paradigmatic examples in what refers to study multibody systems. It contains, as detailed before, four bodies linked by rotational joints. The Figure shows a scheme showing the defined generalized coordinates to express the motion:

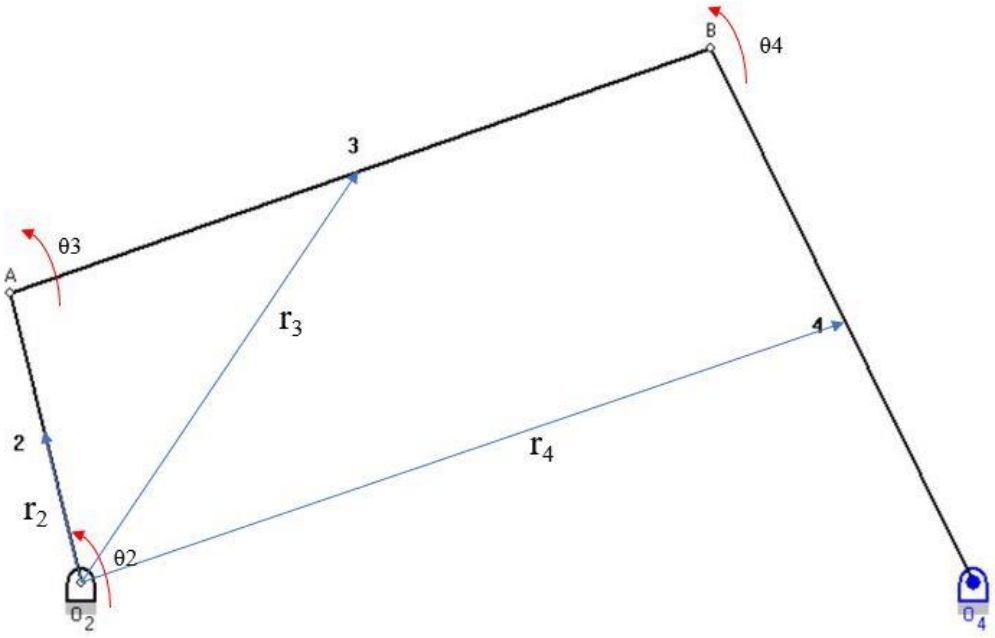


Figure 5.7. 4-bar mechanism built in WinMecC with generalized coordinates indicated [34].

Three generalized coordinates have been defined for each bar (without including the fixed bar): two describing the position of the center of gravity ($\mathbf{r_i} = (x_i\, y_i)$) and one for the orientation (θ_i).

Thus, there are nine generalized coordinates in the model, ordered as shown in the expression below:

$$\boldsymbol{q} = (x_2\ y_2\ \theta_2\ x_3\ y_3\ \theta_3\ x_4\ y_4\ \theta_4)$$

This manner, to calculate them, nine constraints must be defined, eight of them being geometrical constraints:

$$x_2 - (\tfrac{L_2}{2}) * \cos(\theta_2) = 0 \qquad \textbf{(5.4)}$$

$$y_2 - (\tfrac{L_2}{2}) * \sin(\theta_2) = 0 \qquad \textbf{(5.5)}$$

$$x_2 + \left(\tfrac{L_2}{2}\right) * \cos(\theta_2) - x_3 + (\tfrac{L_3}{2}) * \cos(\theta_3) = 0 \qquad \textbf{(5.6)}$$

$$y_2 + \left(\tfrac{L_2}{2}\right) * \sin(\theta_2) - y_3 + (\tfrac{L_3}{2}) * \sin(\theta_3) = 0 \qquad \textbf{(5.7)}$$

$$x_3 + \left(\tfrac{L_3}{2}\right) * \cos(\theta_3) - x_4 + (\tfrac{L_4}{2}) * \cos(\theta_4) = 0 \qquad \textbf{(5.8)}$$

$$y_3 + \left(\frac{L_3}{2}\right) * \sin(\theta_3) - y_4 + (\frac{L_4}{2}) * \sin(\theta_4) = 0 \quad \textbf{(5.9)}$$

$$x_4 + \left(\frac{L_4}{2}\right) * \cos(\theta_4) - L_1 = 0 \quad \textbf{(5.10)}$$

$$y_4 + \left(\frac{L_4}{2}\right) * \sin(\theta_4) = 0 \quad \textbf{(5.11)}$$

While the ninth constraint is the kinematical one, imposing a time restriction to the input angle (in this case, subject to constant velocity ω =-2Π/6 rad/s):

$$\theta_2 - \omega * t = \Pi/2 \quad \textbf{(5.12)}$$

Being L_1, L_2, L_3 and L_4 the respective bar lengths, already defined in paragraph 6.1:

$$L_1 = 1.2\ m$$
$$L_2 = 0.4\ m$$
$$L_3 = 1\ m$$
$$L_4 = 0.8\ m$$

As detailed before, the motion consists of a 360-degrees spin of the input angle. Now, as explained in the mathematical definition, every generalized coordinate can be obtained. In this problem, those which arouse more interest are the rotational coordinates (consult Annex to see their graphic representations).

Now, generalized velocities will be achieved, to that, the jacobian matrix must be calculated:

$$C_q = \begin{pmatrix} 1 & 0 & \left(\frac{L_2}{2}\right) * \sin(\theta_2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\left(\frac{L_2}{2}\right) * \cos(\theta_2) & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -\left(\frac{L_2}{2}\right) * \sin(\theta_2) & -1 & 0 & -\left(\frac{L_3}{2}\right) * \sin(\theta_3) & 0 & 0 & 0 \\ 0 & 1 & \left(\frac{L_2}{2}\right) * \cos(\theta_2) & 0 & 1 & \left(\frac{L_3}{2}\right) * \cos(\theta_3) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\left(\frac{L_3}{2}\right) * \sin(\theta_3) & -1 & 0 & -\left(\frac{L_4}{2}\right) * \sin(\theta_4) \\ 0 & 0 & 0 & 0 & 1 & \left(\frac{L_3}{2}\right) * \cos(\theta_3) & 0 & -1 & \left(\frac{L_4}{2}\right) * \cos(\theta_4) \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\left(\frac{L_4}{2}\right) * \sin(\theta_4) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \left(\frac{L_4}{2}\right) * \cos(\theta_4) \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \textbf{(5.13)}$$

Also, the vector containing the derivatives with respect to time of the kinematical constraints is needed:

$$C_t = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \omega) \quad \textbf{(5.14)}$$

Now, using the Equation 4.3, velocities are obtained. As done with positions, rotational velocities result more important in this research (consult Annex to see their graphics).

Now, to calculate accelerations, the first derivative respect to time of the jacobian matrix is needed:

$$\dot{C}_q = \begin{pmatrix} 0 & 0 & \left(\frac{L_2}{2}\right) * \cos(\theta_2) * \dot{\theta}_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \left(\frac{L_2}{2}\right) * \sin(\theta_2) * \dot{\theta}_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\left(\frac{L_2}{2}\right) * \cos(\theta_2) * \dot{\theta}_2 & 0 & 0 & -\left(\frac{L_3}{2}\right) * \cos(\theta_3) * \dot{\theta}_3 & 0 & 0 & 0 \\ 0 & 0 & -\left(\frac{L_2}{2}\right) * \sin(\theta_2) * \dot{\theta}_2 & 0 & 0 & -\left(\frac{L_3}{2}\right) * \sin(\theta_3) * \dot{\theta}_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\left(\frac{L_3}{2}\right) * \cos(\theta_3) * \dot{\theta}_3 & 0 & 0 & -\left(\frac{L_4}{2}\right) * \cos(\theta_4) * \dot{\theta}_4 \\ 0 & 0 & 0 & 0 & 0 & -\left(\frac{L_3}{2}\right) * \sin(\theta_3) * \dot{\theta}_3 & 0 & 0 & -\left(\frac{L_4}{2}\right) * \sin(\theta_4) * \dot{\theta}_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\left(\frac{L_4}{2}\right) * \cos(\theta_4) * \dot{\theta}_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\left(\frac{L_4}{2}\right) * \sin(\theta_4) * \dot{\theta}_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \textbf{(5.15)}$$

The rotation velocity is constant, so the term associated to the second derivative respect to time of the kinematical constraints is null. This way, the equation 4.5 can be applicated to obtain the accelerations. Those which correspond to rotational accelerations are displayed in Annex.

The kinematic problem has been now solved for the 4-bar mechanism, so the dynamic problem can be solved. Starting with the mass matrix, the mass and the moment of inertia of each body must be defined. Every element has been assigned the same mass:

$$m_1 = m_2 = m_3 = m_4 = 12\ kg$$

With respect to the moment of inertia, the inertia matrix corresponding to an ellipsoid is the following one:

$$I = \frac{1}{5} * m * \begin{pmatrix} b^2 + c^2 & 0 & 0 \\ 0 & a^2 + c^2 & 0 \\ 0 & 0 & a^2 + b^2 \end{pmatrix} \quad \textbf{(5.16)}$$

Being b the longest semiaxis, and a and c the other semiaxis. In this project, b is equal to each body's length (already defined in subchapter 4.2), while a and c are equal to 0.01 m each. As the motion is represented in the OXY plane, only the third term of the inertia matrix results to be of interest. The Table 5.1 contain its value for each body in the four-bar mechanism:

Table 5.1. Moment of inertia of each bar

Body	Input bar	Coupler bar	Output bar
Izz (kg*m^2)	0.08	0.05002	0.03202

Last, the vector containing forces must be defined. In this dynamic problem, only gravitational force was considered, so the vector would be the following one:

$$F = Q_{apl} + Q_v = (0\ -m_2 * g\ 0\ 0\ -m_3 * g\ 0\ 0\ -m_4 * g\ 0) \quad \textbf{(5.17)}$$

Where m_2, m_3 and m_4 are the respective masses of bodies 2, 3 and 4, and g is the gravity, set to its value on Earth: 9,81 m/s^2.

Now, Lagrange multipliers may be obtained according to the equation 4.9. As the mechanism has one degree of freedom, in this case, the input angle θ_2, the objective of the problem is to calculate the motor torque needed to perform the motion. This torque, named M_2 as it is referred to the coordinate θ_2, is associated to the kinematic constraint (the ninth one defined before) so, after applying the Equation 4.10 to obtain generalized forces and moments, the motor torque needed is the one corresponding to the last generalized force (associated to the ninth Lagrange's multiplier):

$$M_2 = -C_q^T * \lambda_9 \quad \textbf{(5.18)}$$

Its graphic representation corresponds to the one displayed in Annex.

The Inverse Dynamics problem is terminated now. The interest now is to calculate the dynamical power generated by the input bar, which is the only one that generates power, for being the only actuator in the model. The Equation serves to obtain this power:

$$P = M * \omega \quad \textbf{(5.19)}$$

Being M the obtained motor torque and ω the angular velocity, described before. This result will be shown and commented in the next chapter.

Once the analytical Inverse Dynamics problem applied to the 4-bar mechanism has been presented, it is the time to approach the problem from the point of view of OpenSim Moco.

5.4 MocoInverse: 4-bar Inverse Dynamics problem in Moco

MocoInverse tool was presented in Chapter 3 as a way to perform Inverse Dynamics problems by solving an

optimal control problem, where the motion obtained in an Inverse Kinematics problem is prescribed. Now, this problem will be particularized for the 4-bar linkage.

In this 4-bar problem, there are no state variables as the systems contains no muscles (later the problem will contain two muscles). As the problem introduced in OpenSim contains 3 generalized coordinates (not including the x and y coordinates for the rotation of the input bar with respect to the bar, and the rotation of the output bar with respect to the bar welded to ground) there will be 3 coordinate actuators which will act as the system controls, and will perform the needed torque to produce the movement so as to fulfill the equations of motion. The used objective function is MocoControlGoal, which minimizes the squared weighted sum of controls integrated through time (in this case, the coordinate actuators) [35]. This objective function is defined as follows [13]:

$$f = \frac{1}{d} * \int_{t0}^{tf} \sum_{c=1}^{N} w_c * |x_c(t)|^2 dt \qquad \textbf{(5.20)}$$

Where d is the displacement of the system (it gets assigned a value of 1), t_0 and t_f are the initial and final time instants of the cycle (in this work, $t_0 = 0$ s and $t_f = 6$ s), N is the number of controls (in this case, there are 6 coordinate actuators, so N=6), and w_c represents each control variable.

The restrictions to the optimization problem include:

a) The vector of generalized accelerations, u, obtained as the second derivative of the generalized coordinates vector [13]:

$$u = \ddot{q} \qquad \textbf{(5.21)}$$

b) The Euler-Newton equations of motion [13]:

$$M * u + G^T * \lambda = f_{app} + f_{grav} \qquad \textbf{(5.22)}$$

Where M is the mass matrix, G^T is the transpose jacobian matrix of constraints, λ is the vector containing Lagrange multipliers, f_{app} is the vector of applied forces and f_{grav} the vector of gravitational forces.

c) Position-level constraints [13]. Every restriction to join the bodies in the model is considered, including:
c.1) The ball joint to link the ground and the input bar.

c.2) The three pin joints to join the input with the coupler bar, the coupler with the output bar and the output bar with the fifth bar, which is welded to the ground.

c.3) The two point constraints used to join the fifth bar to the ground.

They are represented in a vector containing equations equalled to zero:

$$\Phi(q) = 0 \qquad \textbf{(5.23)}$$

Being q the vector containing generalized coordinates.

Finally, bounds must be set. In this project, the coordinate actuators have no defined bounds (those are -∞ and ∞), so the formulation of the MocoInverse problem can be written this way:

$$Min \qquad \int_{t_0}^{t_f} \sum_{c=1}^{6} w_c * |x_c(t)|^2 dt$$

$$subject\ to \qquad u = \ddot{q}$$

$$M * u + G^T * \lambda = f_{app} + f_{grav}$$

$$\Phi(q) = 0$$

$$with\ respect\ to \qquad t_0 = 0\ s;\ \ t_f = 6\ s$$

Once completely defined the MocoInverse problem for the 4-bar mechanism, results can be studied. As there are no state variables in this problem, the only results will be the Coordinate Actuators, it is said, the control variables. Each of the 3 Coordinate Actuators corresponding to the 3 angles between the ground, the input, the coupler and the output bars develop a motor torque that is shown in the Figures, present in the Annex.

Now, the power developed by the total set of Coordinate Actuators can be calculated, it must coincide with the power calculated by the analytical way. To obtain this total power, this equation must be used [32]:

$$Power = \sum_{i=1}^{6} Motor\ torque_i * \omega_i \qquad \textbf{(5.24)}$$

Being ω_i the angular velocity for each angle and corresponding $Motor\ torque_i$ with torques generated by each of the six Coordinate Actuators in the problem. This expression is also valid to calculate the power in the Inverse Dynamics made in OpenSim GUI.

Now the Inverse Dynamics problem in Moco has been solved for the 4-bar mechanism, it is the moment to present the muscled 4-bar, which will represent the union between the human being and the exoskeleton.

5.5 Muscled 4-bar mechanism

This linkage will definitely dilucidate if OpenSim Moco can be used to study dynamics in a closed-loop biomechanical model, especially a human being wearing an exoskeleton. Two muscles have been added to the original 4-bar device, each one at one side of the input bar, as it can be seen in Figure:

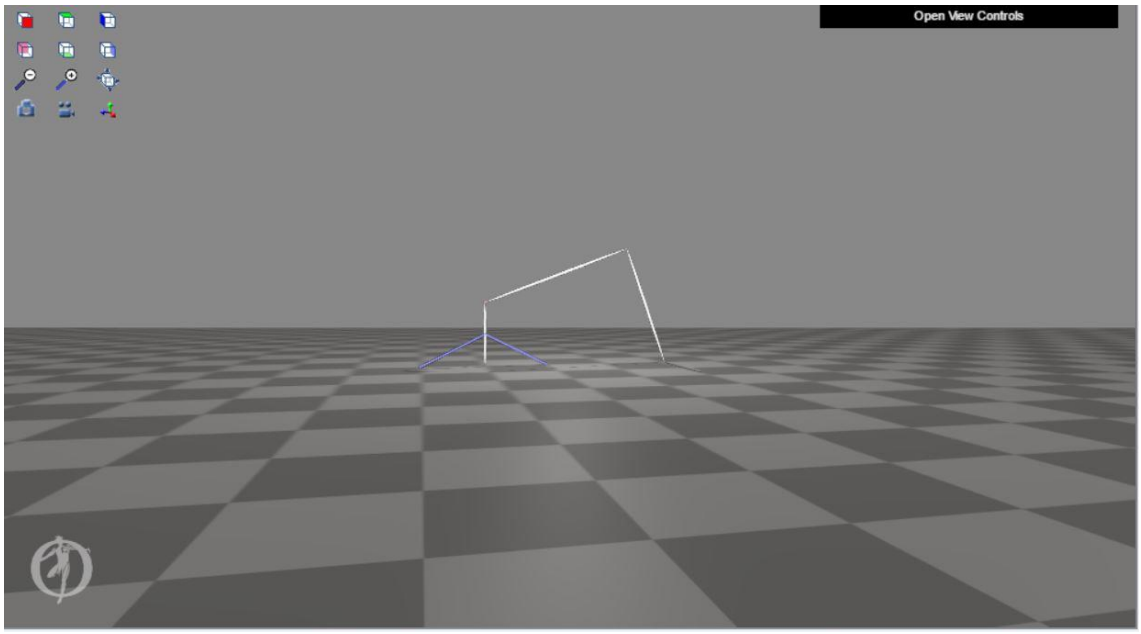


Figure 5.8. 4-bar linkage with two muscles, each at one side. OpenSim [6]

Both muscles have a common insertion point, located in the center of mass of the input bar, while the other insertion points have been placed symmetrically from each other, with respect to the initial point of the input bar, at 0.2 m from this point. The common insertion point will later be changed so as to study the influence of this variation in the torque generated by the muscle, and so on the meaning this could have in a real exoskeleton-human model.

Muscles do not intervene in kinematics, as they do not introduce any degree of freedom nor coordinate; neither do they will change dynamics in the problem, because their mass and inertia will be considered negligible in this project. This means that the required torque to solve the problem remains the same than in the problem without muscles.

They constitute two new actuators whose mission is to perform by themselves the required torque to make the desired motion (the 360-degrees spin of the input bar). If during the movement they are not able to generate this torque, Moco will activate Coordinate Actuators to produce this missing torque. In reality, this will mean that these muscles cannot make the motion by themselves, and they need an external device.

The muscle model used to introduce these muscles was Thelen's model. The Table 5.2 collects all characteristic values needed to define each muscle (both muscles are exactly the same):

Table 5.2. Main muscle properties. Data from OpenSim muscles [6].

Max. isometrical force (N)	Optimal fiber length (m)	Tendon slack length (m)	Pennation angle (optimal fiberlengths/s)	Max. contraction velocity (optimal fiberlengths/s)
162	0.352	0.126	0.05235988	10

However, for the optimization, MocoInverse changes the muscle model to De Groote & Fregly's [19]. The reason consists of the active force-length-velocity curves described by these model: they are more suitable for the optimization as they contain less discontinuities, and are normally twice-derivable in the whole graphic [19].

Now, Inverse Dynamics will be particularized for the muscled 4-bar, with results appearing in Chapter 7 (and intermediate results in Annex).

5.6 Inverse dynamics in a muscled 4-bar

Even when MocoInverse will still be used to perform the Inverse Dynamics problem, the presence of muscles will complicate the problem as each muscle introduces two state variables: the normalized tendon force and the activation, as well as one control variable: the muscle excitation.

This way, a new term must be introduced in the objective function to compute these new states. This term is the MocoSumSquaredStatesGoal [36], and it minimizes the squared weighted sum of states (in this case, the activation and the normalized tendon force). It is expressed this way:

$$f = \int_{t_0}^{t_f} \sum_{s=S} w_s * y_s(t)^2 dt \qquad \textbf{(5.25)}$$

Where $y_s(t)$ correspond to the state variable s, and w_s is its weight (by default, analogously to controls, every weight will be 1). This objective function is added to the one previously added to the problem without muscles, MocoControlGoal. Moreover, an end-point constraint must be added: MocoInitialActivationGoal [37].

$$a_{i,t0} = a_{i,tf} \qquad \text{i=1...number of muscles} \qquad \textbf{(5.26)}$$

This constraint forces the initial and the final activation to be the same, what is compulsory as the problem is cyclical: the motion ends at the same point where it began.

Last, the new introduced variables are bounded. Activations are bounded between 0 and 1, normalized tendon forces are limited between 0 and 5, while muscle excitations' bounds are 0 and 1:

$$0 \leq a_i \leq 1 \qquad \text{i=1...number of muscles}$$

$$0 \leq \widetilde{F_{tendon}} \leq 5$$

$$0 \leq u \leq 1$$

The rest of the constraints remain the same as the previous MocoInverse problem, without muscles. Thus, the formulation of the MocoInverse problem for the muscled 4-bar would be the one shown right below:

$$Min \qquad \int_{t_0}^{t_f} \sum_{c=1}^{6} w_c * |x_c(t)|^2 dt \quad + \int_{t_0}^{t_f} \sum_{s=S} w_s * y_s(t)^2 dt$$

$subject\ to$ $\quad u = \ddot{q}$

$$M * u + G^T * \lambda = f_{app}$$

$$\Phi(q) = 0$$

$$a_{i,t0} = a_{i,tf}$$

$with\ respect\ to$ $\quad t_0 = 0\ s;\ \ t_f = 6\ s$

$0 \leq a_i \leq 1$ $\quad$ i=1...number of muscles

$$0 \leq \widetilde{F_{tendon}} \leq 5$$

$$0 \leq u \leq 1$$

In this problem, muscle forces and torques generated by Coordinate Actuators have been calculated. Thus, the next step is to obtain the power. The Equation 6. is still the way to get the power generated by Coordinate Actuators. Regarding muscle power, it must be obtained following the next equation [32]:

$$P_{muscles} = |F| * |v| * \cos(\gamma) \qquad \textbf{(5.27)}$$

Being F the total muscle force, v the linear velocity of the center of gravity of the input bar, and γ the angle conformed by those vectors. The linear velocity is defined this way:

$$v = \omega * \frac{L_2}{2} \qquad \textbf{(5.28)}$$

With $\frac{L_2}{2}$ being the radius of rotation, and ω the angular velocity. To define γ, firstly it is necessary to introduce α and β, which are the angles between the ground and the right and left muscles, respectively:

$$\alpha = \text{atan}\left(\frac{0.2*\sin\left(\frac{\Pi}{2}-\theta_2\right)}{0.4+0.2*\cos\left(\frac{\Pi}{2}-\theta_2\right)}\right) \quad \textbf{(5.29)}$$

$$\beta = \text{atan}\left(\frac{0.2*\sin\left(\frac{\Pi}{2}-\theta_2\right)}{0.4-0.2*\cos\left(\frac{\Pi}{2}-\theta_2\right)}\right) \quad \textbf{(5.30)}$$

In this moment the angles between the muscles and the ground, γ_1 for the right muscle, and γ_2 for the left muscle, can be defined:

$$\gamma_1 = \Pi - (\alpha + \theta_2) \quad \textbf{(5.31)}$$

$$\gamma_2 = \beta - \theta_2 \quad \textbf{(5.32)}$$

Now, the power can be analyzed, comparing the total with the one muscles have been able to generate, and the one which had to be exerted by the virtual Coordinate Actuators.

This manner, every formulation required to perform Inverse Dynamics has been introduced. Results may be displayed and commentated.

6 RESULTS

In this chapter, results which confirm hypotheses to prove in this work will be shown. To arrive to them, several intermediate results were necessary; these appear in the Annex.

6.1 Analytical Inverse Dynamics

The first problem to solve was the analytical Inverse Dynamics in the non-muscled 4-bar, introduced in this work in paragraph 5.6. As explained then, the objective was to calculate the motor torque required to perform the motion, associated to the only degree of freedom in the mechanism, the input angle θ_2. The result is displayed right below:

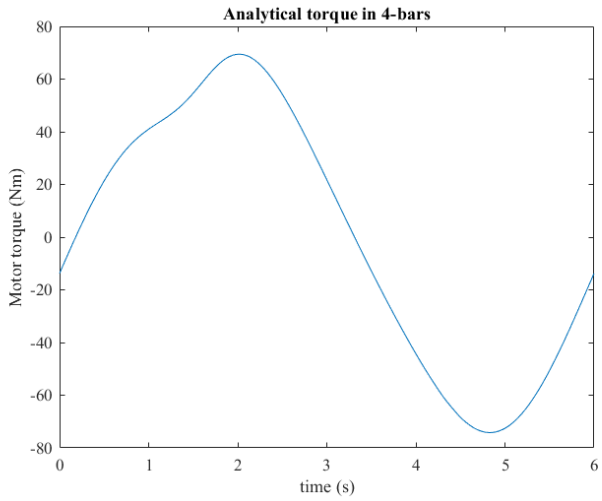


Figure 6.1. Analytical motor torque of 4-bar. Obtained in Matlab.

Regarding the power, this was the result calculated:

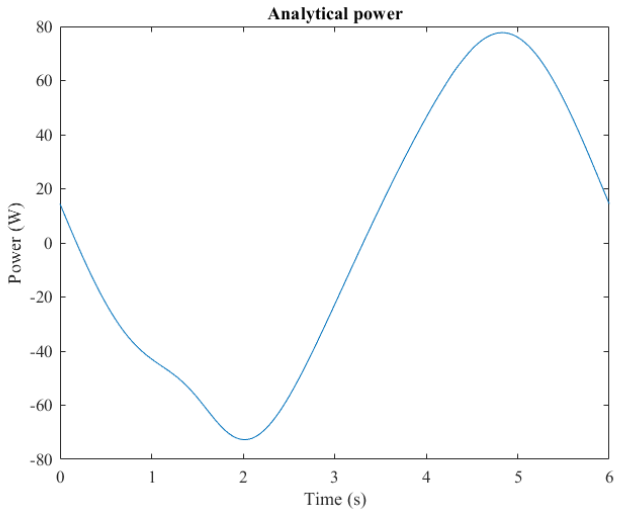


Figure 6.2. Analytical power of 4-bar. Obtained in Matlab

Being this result of great importance in this work as it has served as the reference to compare the diverse values of power obtained with, so as to check the validity of methods to perform Inverse Dynamics.

Observing the curve in Figure 6.2, and as expected knowing the two phases in which the motion is divided, also two phases can be clearly appreciated: a first one, spanning until the half of the cycle, where the power is negative, and a second one, parting from the time instant t = 3 s, where the power is positive. In the first half, the input is rotating downwards, so its objective is to oppose this fall, and exerts a negative power with this purpose. In the second phase, the mechanism is at its slowest position, and it needs to generate a positive power to raise the bar and turn it to the initial place.

6.2 Impossibility to use ID in closed-loop mechanisms in OpenSim GUI

The first circumstance required to check was the impossibility to obtain correct results from the Inverse Dynamics performed in the Graphical User Interface of Moco in a closed-loop mechanism. This can be checked by comparing the dynamical equivalent power obtained in this problem to the power resulting in the analytical problem. These graphics are shown in Figures 6.3 and 6.4, respectively:

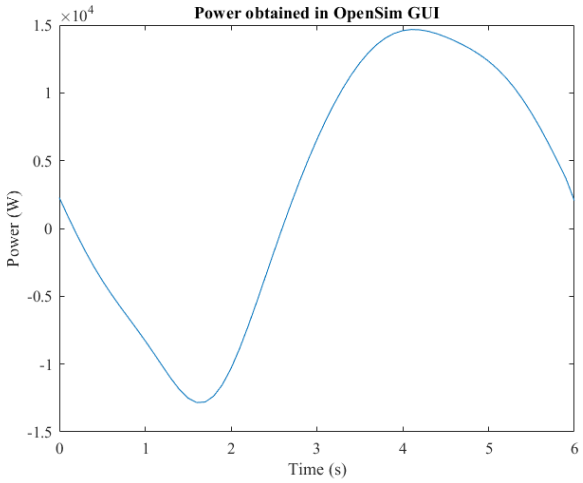


Figure 6.3. Power resulting in ID by OpenSim. Obtained in Matlab.

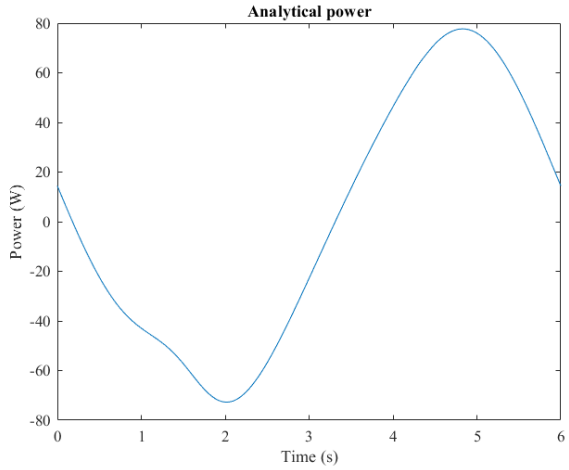


Figure 6.4. Analytical power of 4-bar. Obtained in Matlab

It is clearly seen that these powers do not coincide, what verificates that OpenSim GUI is not a valid tool to perform Inverse Dynamics. This circumstance was waited, because the reason why OpenSim cannot solve Inverse Dynamics in closed-loop mechanisms is that it does not recognize constraints in the dynamical problem, what leads to errors at the moment of calculating forces and moments.

6.3 Capacity of OpenSim Moco to solve ID in closed-loop mechanisms

The next step consisted of determinate if the problem solved by using MocoInverse, in the closed-loop mechanism (in this case, the 4-bar without muscles) offered correct and reliable results, it is said, the power generated in this problem might be the same as in the analytical one. Figure 6.5 displays the power generated by the 4-bar mechanism without muscles, and Figure 6.6 compares it to the obtained in the analytical case:

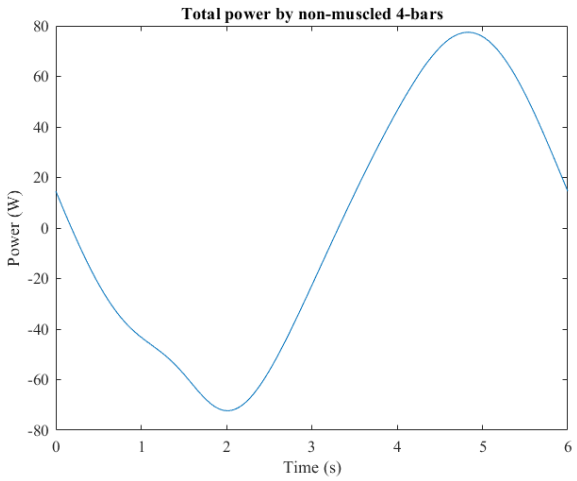


Figure 6.5. Power in a non-muscled 4-bar mechanism. Obtained in Matlab.

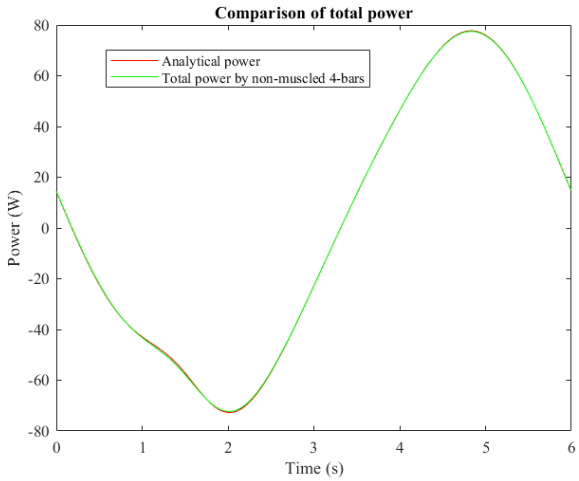


Figure 6.6. Comparison of total power obtained in Moco and the analytical way. Obtained in Matlab.

Results confirm what was being tried to be proven: the power generated by the 4-bar is the same calculated by the analytical way and by using Moco. This confirms the validity of this tool to perform Inverse Dynamics analysis in closed-chain mechanisms.

6.4 Capacity of OpenSim Moco to solve ID in muscled closed-loop mechanisms

Now it is the moment to evaluate if OpenSim Moco can serve to study closed-loop mechanisms with muscles, assimilable to models involving human people wearing exoskeletons. The requirement is still that the power supplied by the muscled 4-bar were the same as in the model without muscles, because, as explained before, muscles play the role of actuators, providing force (and thus, torque and power), but the motion is still the same: the input bar must complete a 360-degrees rotation, and the analytical solution informs about the required torque to fulfill this movement.

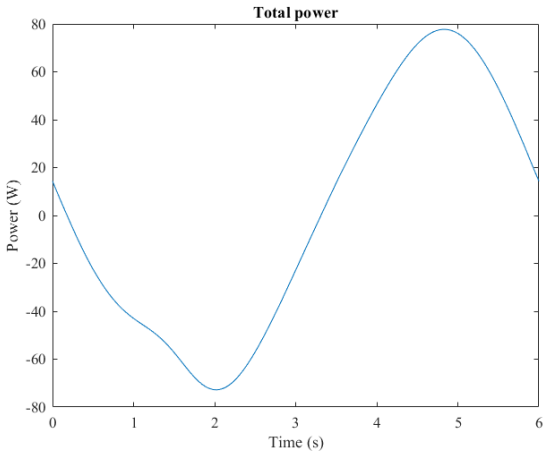


Figure 6.7. Power resulted in muscled 4-bar. Obtained in Matlab.

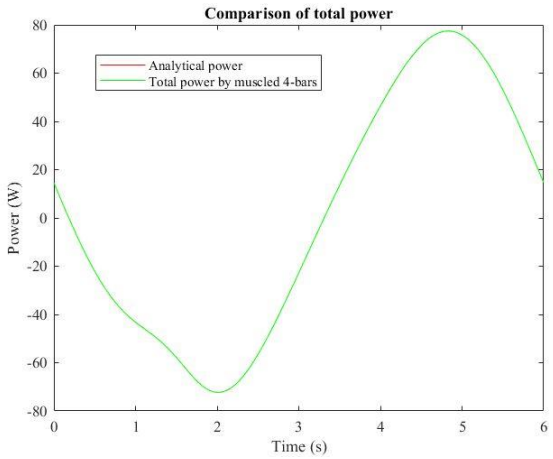


Figure 6.8. Comparison of total power obtained in Moco and the analytical way, in the muscled 4-bar. Obtained in Matlab.

In Figure 6.7, power supplied by the muscled 4-bar appears, while in Figure 6.8 it is compared to the analytical one.

It is clear that, effectively, the power calculated in the problem involving the muscled 4-bar mechanism is the same as the one obtained in the analytical problem. Therefore, OpenSim has been confirmed as an accurate tool to exert Inverse Dynamics problems in closed-loop musculoskeletal models, in which problems involving exoskeletons are, as previously said in this work, included.

But analysis of results must continue yet. Now, the influence of muscles in the motion will be detailed, focusing on their activations, the forces they generate, both passive and active, and how they contribute to the total power in contrast to the Coordinate Actuators.

In first place, the contribution of these Coordinate Actuators and the muscles to the total power is analyzed in Figure 6.9:

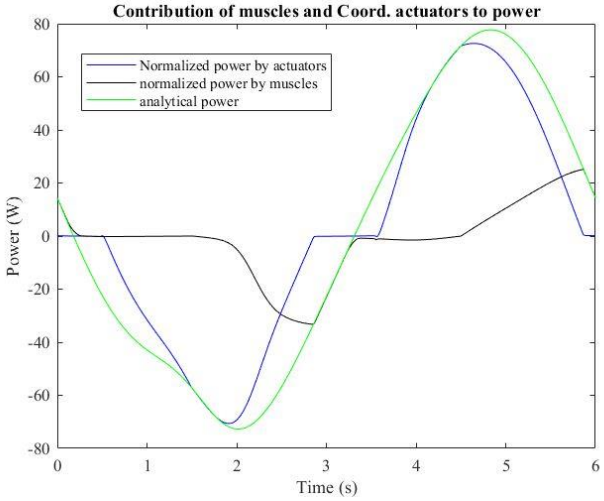


Figure 6.9. Contribution of muscles and Coord. Actuators to power. Obtained in Matlab.

It is observed that the muscles (whose power is blacklined) cannot generate the whole needed power to perform the motion. In some parts of the cycle, they exert the required part (from t = 0 to t = 0.5 s and from t = 3 s to t = 5.7 s. During the rest of time, Coordinate Actuators have to supply power too, and even in two instants (t = 1.5 s and t = 4.5 s) these actuators provide the 100% of the power. The explanation to that resides on the fact that the input bar is aligned to the ground in those instants, so the moment arm to generate the torque is zero. Muscles cannot therefore produce torque nor power.

The appearance of the Coordinate Actuators in the motion is directly related to muscle activation. It can be observed in Figure, where powers have been normalized so as to focus on the evolution of the graphic and not in magnitude orders:

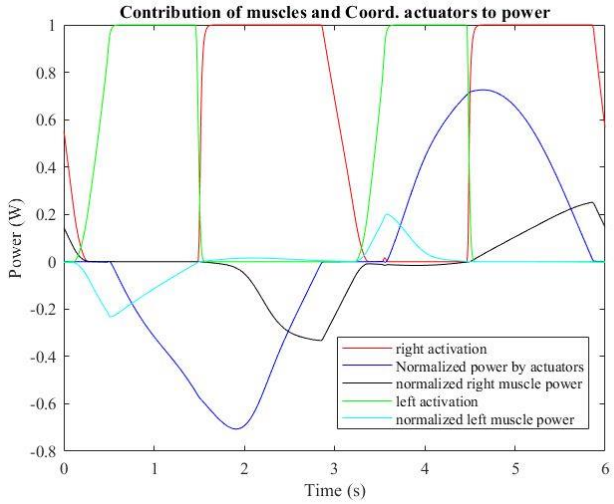


Figure 6.10. Comparison of powers by muscle and their activations. Obtained in Matlab

It is observed that muscles are able to produce the whole force required until they reach their maximum level of activation. This means they cannot generate more force, and so, the Coordinate Actuators start to supply torque. When muscle activation is under 1, muscles are generating all the force needed, and they do not need to get more activated.

In order to study the relation of the activation with respect to the active and passive forces, as well as the muscle parameters intervening in the force (fiber length and contraction velocity) the next Figures will be useful:

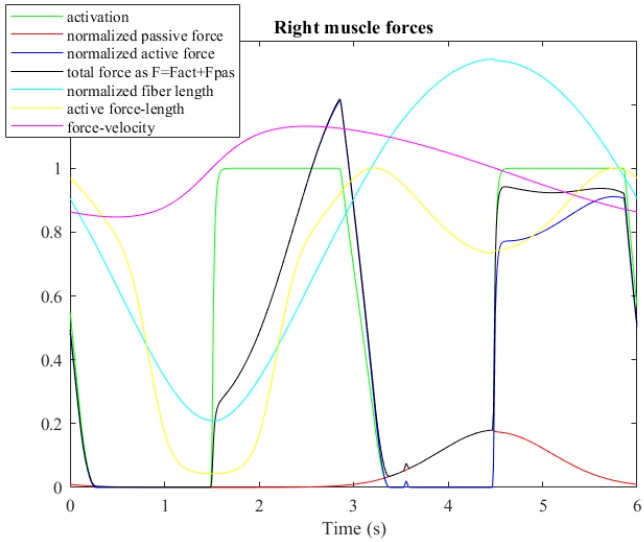


Figure 6.11. Right muscle forces. Obtained in Matlab

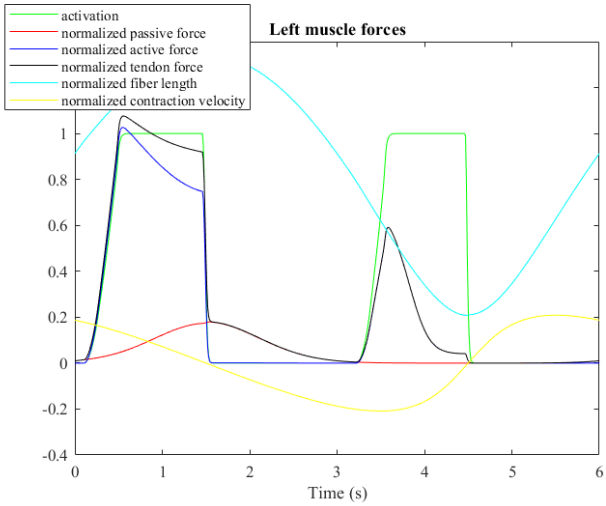


Figure 6.12. Left muscle forces. Obtained in Matlab

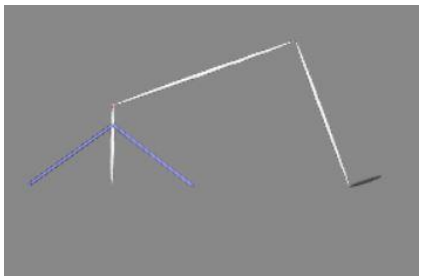
It can be observed in both muscles that passive forces appear when the fiber lengths surpass its optimal fiber length. This can happen even if the muscle is not activated, for example, as it happens in the left muscle between the instants t =1.5 s and t = 3.3 s. The muscle is shortening, but it will be producing passive forces while its fiber length is over its optimal length.

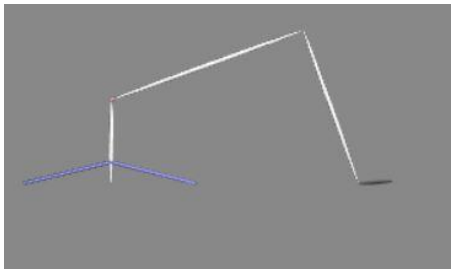
On the other hand, there is no active force if the activation is zero, as expected. But the value of the active force also depends on the active force-length-velocity parameters, as it happens in right muscle between instants t =4 s and t=4.5 s, when the active force grows up since these parameters are growing up. When the fiber length gets closer to its optimal length, higher is the value of the active force-length parameter, the same occurs with force velocity parameter when the contraction velocity gets increased. But when fiber length is too much bigger or smaller than its optimal length, or contraction velocity is too short (or even negative, what means that muscle is lengthening), the active force length gets decreased and so, the muscle can generate less power.

To sum up, it can be said that muscles stop generating the force required by the problem when their activation reaches its maximum value, and consequently, Coordinate Actuators supply power. Furthermore, if the fiber length is far from its optimal value, and contraction velocity is small (or negative), the production of force by the muscle will be limited. This situation can be improved if the position of the muscle is changed, what will be seen in the next paragraph.

6.5 Influence on power of changes in the insertion point of muscles

Last, to close this work, a sensitivity analysis has been done to study what happens to muscle power if the insertion point is changed. This way, the insertion points of muscles in the input bar have been modified the way shown in Figures 6.13 and 6.14:





Figures 6.13 and 6.14. New insertion points of muscles in input bar. OpenSim [6]

In the first case, the insertion point common to both muscles has been placed above the center of gravity of the input bar; concretely, in the middle point between the center of gravity and the vertix common to the input and the coupler bar; in the second case, the insertion point has been collocated between the center of gravity and the vertix of the input bar in contact to the ground. Then, in the third phase of this analysis of sensitivity, the two muscles inserted in the center of gravity of the input bar have been maintained, and the two muscles situated above have been have been considered, as shown in Figure 6.15:

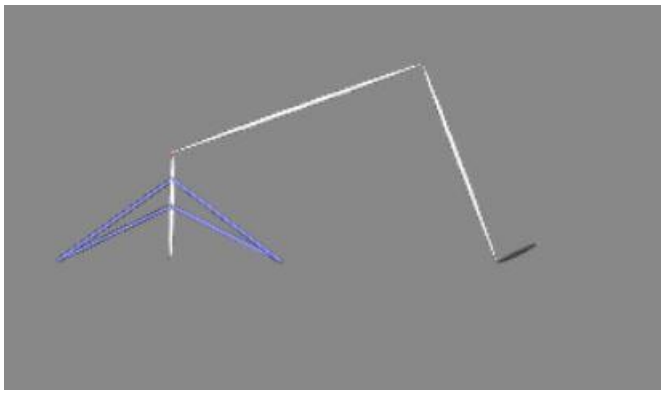


Figure 6.15. 4-muscles 4-bar linkage in OpenSim [6]

Results of this analysis of sensitivity are now shown. Starting with the study of influence of the collocation of

the insertion point, the graphics corresponding to the power generated by muscles is displayed in Figure 6.16, while the one performed by the actuators can be observed in Figure 6.17:

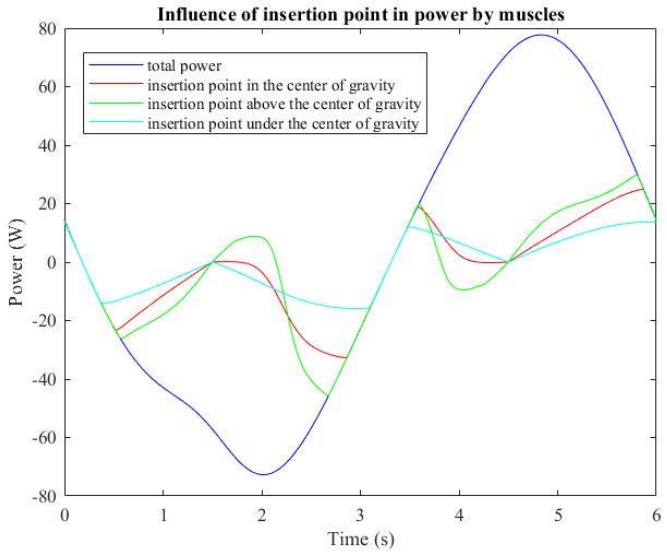


Figure 6.16. Influence of insertion point in muscle power. Obtained by Matlab.

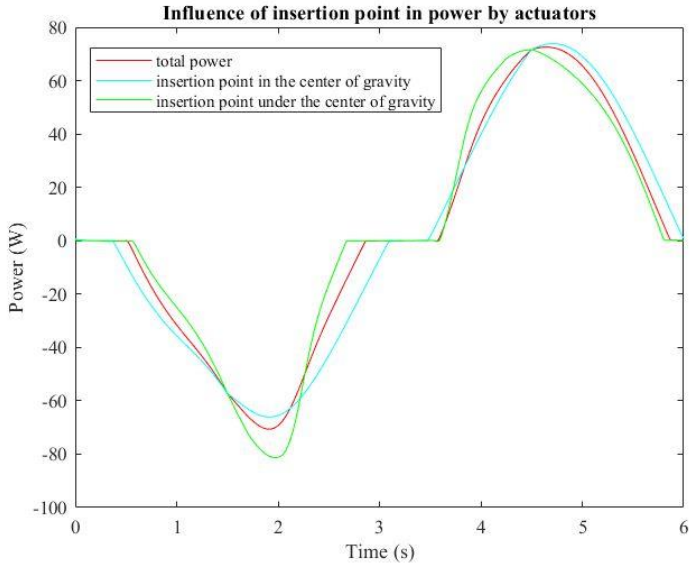


Figure 6.17. Influence of insertion point in actuators' power. Obtained by Matlab.

It can be seen that locating the insertion point above the middle point of the input bar (this case has been represented by the green line) results favorable during most of the cycle in terms of resemblance of muscle power to total power, because the power generated by the muscles equals the power required to exert the motion during more time than in the case where the insertion point is in the center of gravity of the bar. As the power generated by Coordinate Actuators is the resulting from subtract muscle power from total, the power from actuators is zero during these instants of time. This result can seem logical, because at placing the insertion point this way, the moment arm gets increased, so it is easier for the muscle to generate more torque.

However, this circumstance does not always occur in the whole cycle. In trams between t = 1.5 s and t =2.3 s, and between t = 3.5 s and t =4.5 s, the option to place the insertion point under the center of gravity of the bar makes the muscle to perform a power nearer of analytical than in the other option. It might have to do with the particular geometry in those instants of time: the orientation of muscles might favor to place them under the center of gravity, but as observed in the cycle, generally it results better to choose the first option: to increase the moment arm of muscles.

Now, concerning the case with 4 muscles, a bright result is obtained: the power generated by muscles is slightly the same as the needed to perform the whole rotation of the input bar. The inclusion of two additional muscles, in this case, by inserting them further away than the connection to the ground, has reduced to the minimum the power generated by actuators. This was expected as a bigger presence of muscles could make logical a bigger muscle power. This result is shown in Figure 6.18.

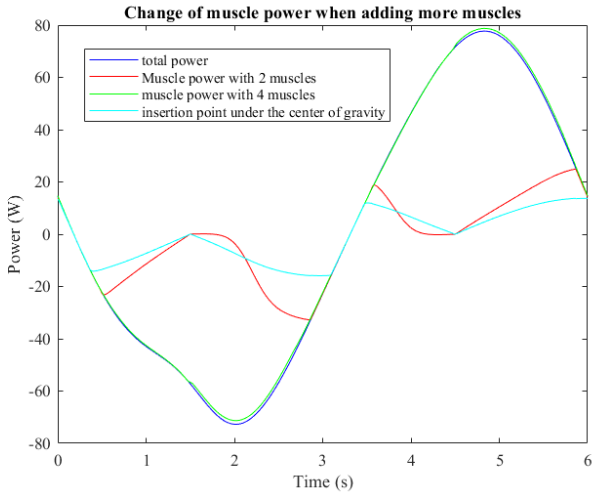


Figure 6.18. Change of muscle power when adding more muscles. Obtained in Matlab.

7 CONCLUSIONS. LOOKING TO FUTURE

This work parted with the objective to evaluate the possible implementation of a method to study dynamics in musculoskeletal models wearing exoskeletons, and this purpose can be considered fulfilled. For that, it has been necessary to make various simplifications, considering the most basic model of closed-loop mechanism, the 4-bar mechanism, with some muscles attached to its input bar, but the results obtained prove that OpenSim Moco can be utilized to study more complex systems, containing more muscles, more coordinates and specially, more degrees of freedom. The direct collocation methods employed by Moco to solve the optimal control problem are proven to lighten up the process to find a solution, what also will serve to accelerate investigations related to this field.

Moreover, the collocation of the muscle in the model has also been demonstrated to be important due to its influence in the power generated by muscles. This circumstance must be considered at designing exoskeletons, specially what concerns the insertion between the device and the human individual. An optimal interaction between the musculoskeletal model and the mechanical device will probably result decisive to improve the performance of the biomechanical model in terms of optimality (e. g. minimizing the excitation effort), or allowing muscles to perform more torque generating less force (as the insertion exerts an influence in the moment arm).

Implications of results obtained in this work could result slightly notable if further research is being made in the future. The musculoskeletal human model is a really complex subject to study due to the quantity of bodies, muscles and joints it contains, and it becomes even more complicated when considering a mechanical device like an exoskeleton, which is also composed of bodies, joints and actuators, and on top of that, conform a closed-loop biomechanical model. The possibility to use the methodology described in this research could make slightly easier future studies about the implementation of exoskeletons in human people, as the optimal control problem allows to solve, in a relatively short amount of time, dynamic problems which minimize magnitudes of importance in the motion performed by the person subject to study.

Ultimately, it must not be forgotten the importance exoskeletons are starting to have in our society, and will progressively have in the future, as they are being reveled as devices which make more comfortable our lives, either supplying a support to perform some motions (since carrying loads to even walking) to those people who have suffered several injuries which may partially incapacitate them, either helping industrial workers to perform their duties reducing their fatigue and muscle excitation effort during their workday.

But none of the conclusions citated above will serve if further investigations are not done. This project must continue as it has been done using a simplified musculoskeletal model, so research must follow by studying more complex models. Patience, always required to perform any type of investigation, will be worth it as the evolution in this field of study will undoubtedly improve human people's lives.

8 BIBLIOGRAPHY

[1] RoboticsBiz, Everything about Robotics and AI, *Active vs Passive Exoskeletons explained*. October 2021. Available online: https://roboticsbiz.com/active-vs-passive-exoskeletons-explained/

[2] Wikipedia, the free encyclopedia. *Powered exoskeleton,* November 2021. Available online: https://en.wikipedia.org/wiki/Powered_exoskeleton

[3] Theurel, J., Desbrosses, K., Roux, T., Savescu, A. 2018. Physiological consequences of using an upper limb exoskeleton during manual handling tasks. Applied Ergonomics. 67, pp. 211–217.

[4] Yale University, *Active Orthosis and Exoskeletons*, New Haven, Connecticut, 2012. Available online: https://www.eng.yale.edu/grablab/research_orthoses.html

[5] Brian Umberger, Nicholas Bianco, Thomas Uchida, Christopher Dembia. *OpenSim Moco*, Stanford, California, 2019.

[6] Nicholas Bianco, Thomas Uchida, Christopher Dembia. *OpenSim*, Stanford, California, 2007.

[7] Bueno, D. R. and Montano, L., *Multijoint upper limb torque estimation from sEMG measurements,* 35th Annual International Conference of the IEEE, Osaka, Japan, July 2013.

[8] Sartori, M., Reggiani, Lloyd, D.G., Pagello, E, *A neuromusculoskeletal model of the human lower limb: towards EMG-driven actuation of multiple joints in powered orthoses,* IEEE International Conference on Rehabilitation Robotics, Zurich, Switzerland, June 2011.

[9] Maza Ortega, D., *Análisis dinámico inverso de la marcha humana,* Universidad de Sevilla, Seville, Spain, 2020.

[10] Font-Llagunes, J.M., García-Vallejo, D., Schiehlen, W, *Dynamical analysis and design of active orthoses for spinal cord injured subjects by aesthetic and energetic optimization,* Nonlinear Dynamics, 2014.

[11] Rina García, N., *Development of an Optimal Control Framework to Predict Human Motion,* Universitat Politècnica de Catalunya, June 2020.

[12] OpenSimVideos in Youtube, *Webinar - OpenSim Moco: Software to optimize the motion and control of OpenSim models,* Stanford University, Stanford, California, November 2019. Available online:

https://www.youtube.com/watch?v=IYYZgyE33pU&t=88s

[13] OpenSim Moco, *OpenSim Moco Documentation*. October 2021. Available online: https://opensim-org.github.io/opensim-moco-site/docs/

[14] Rodríguez Uría, M.V., López Fernández, M.Á., Pérez Gladish, B.M., *Aplicaciones económicas del control óptimo. El problema de la maximización de la utilidad individual del consumo. El problema del mantenimiento y momento de venta de una máquina.* Facultad de Ciencias Económicas, Universidad de Oviedo. 1996.

[15] Kelly, M., *An introduction to trajectory optimization: how to do your own direct collocation.* SIAM Review, 59(4), 849-904. 2017.

[16] Nicholas Bianco, Thomas Uchida, Christopher Dembia, *OpenSim Documentation.* Stanford University. 2021. https://simtk-confluence.stanford.edu:8443/display/OpenSim/Documentation

[17] Nicholas Bianco, Thomas Uchida, Christopher Dembia, *OpenSim Documentation. 2003 Thelen Muscle Model.* Stanford University. 2021. https://simtk-confluence.stanford.edu:8443/display/OpenSim/Thelen+2003+Muscle+Model

[18] Thelen, D. G., *Adjustment of muscle mechanics model parameters to simulate dynamic contractures in older adults* . Journal of Biomechanical Engineering. University of Wisconsin. February 2003.

[19] De Groote, F, Kinney, A. L., Rao, A. V., Fregly, B. J., *Evaluation of Direct Collocation Optimal Control Problem Formulations for Solving the Muscle Redundancy Problem.* Annuals of Biomedical Engineering, Vol. 44, nº 10. October 2016.

[20] Nicholas Bianco, Thomas Uchida, Christopher Dembia, *OpenSim Documentation. OpenSim Models.* Stanford University. 2021. https://simtk-confluence.stanford.edu:8443/display/OpenSim/OpenSim+Models

[21] Martínez Reina, J. *Comportamiento mecánico del músculo esquelético*. Bioengineering, Universidad de Sevilla, 2018.

[22] Ojeda Granja, J. *Fisiología y Comportamiento del tejido muscular*., Universidad de Sevilla, 2021.

[23] Hill, A. V. *The heat of shortening and the dynamic constant of muscle.* Royal Society. October 1938.

[24] Ojeda Granja, J. *Application of multibody system techniques to locomotor system.* Universidad de Sevilla. 2012.

[25] García Vallejo, D. *Three-Dimensional Simulation of Human Walking Optimizing Aesthetics and Energy*, University of Stuttgart, 2010.

[26] Nagano, A., Gerritsen, K.G.M.: *Effects of neuromuscular strength training on vertical jumping performance—a computer simulation study*. J. Appl. Biomech. 17, 113–128 (2001).

[27] OpenSim, *OpenSim Documentation*. *First Order Activation Dynamics.* October 2021. Available online: https://simtk-confluence.stanford.edu:8443/display/OpenSim/First-Order+Activation+Dynamics

[28] OpenSim, *OpenSim Documentation*. *Characteristic Musculotendon curves.* October 2021. Available online: https://simtk-confluence.stanford.edu:8443/display/OpenSim/Characteristic+Musculotendon+Curves

[29] OpenSim, *OpenSim Documentation*. *How Inverse Kinematics works.* October 2021. Available online: https://simtk-confluence.stanford.edu:8443/display/OpenSim/How+Inverse+Kinematics+Works

[30] OpenSim, *OpenSim Documentation*. *How Inverse Dynamics works.* October 2021. Available online: https://simtk-confluence.stanford.edu:8443/display/OpenSim/How+Inverse+Dynamics+Works

[31]. Exoskeleton model in OpenSim. Departamento de Ingeniería Mecánica y Fabricación. Universidad de Sevilla. 2020.

[32]. Chamorro Moreno, R. et al., *Teoría de Máquinas y Mecanismos.* Departamento de Ingeniería Mecánica y Fabricación. Universidad de Sevilla. 2015.

[33]. Escalona Franco, J.L., *Cinemática y Dinámica de Máquinas.* . Departamento de Ingeniería Mecánica y Fabricación. Universidad de Sevilla. 2015.

[34]. WinMecc. Área de Ingeniería Mecánica de la Universidad de Málaga.

[35] OpenSim Moco, *OpenSim::MocoControlGoal.* October 2021. Available online: https://opensim-org.github.io/opensim-moco-site/docs/

[36] OpenSim Moco, *OpenSim::MocoSunSquaredStateGoal.* October 2021. Available online: https://opensim-org.github.io/opensim-moco-site/docs/

[37] OpenSim Moco, *OpenSim::MocoInitialActivationGoal.* October 2021. Available online: https://opensim-org.github.io/opensim-moco-site/docs/

ANNEX. INTERMEDIATE RESULTS

In this annex, some intermediate results without direct relationship with the purpose of this work are mostrated. They are several muscle parameters obtained during the process to calculate passive and active forces (described in Chapter 4).

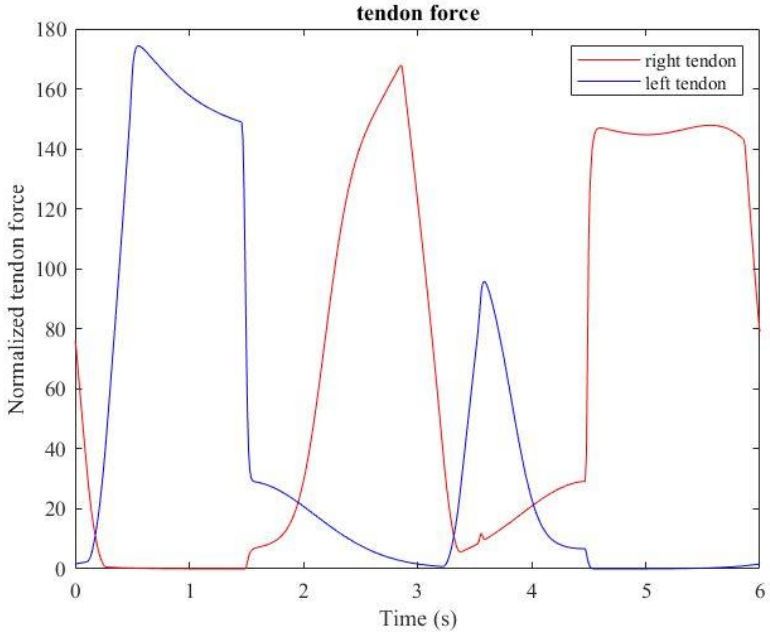


Figure 9.1. Normalized tendon force in both muscles. OpenSim Moco [13]

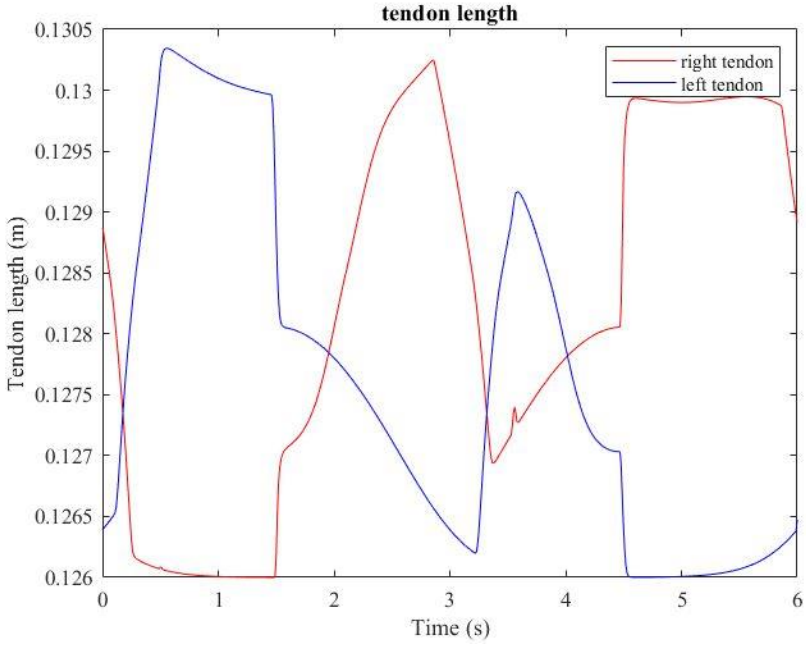


Figure 9.2. Normalized tendon force in both muscles. OpenSim Moco [13]

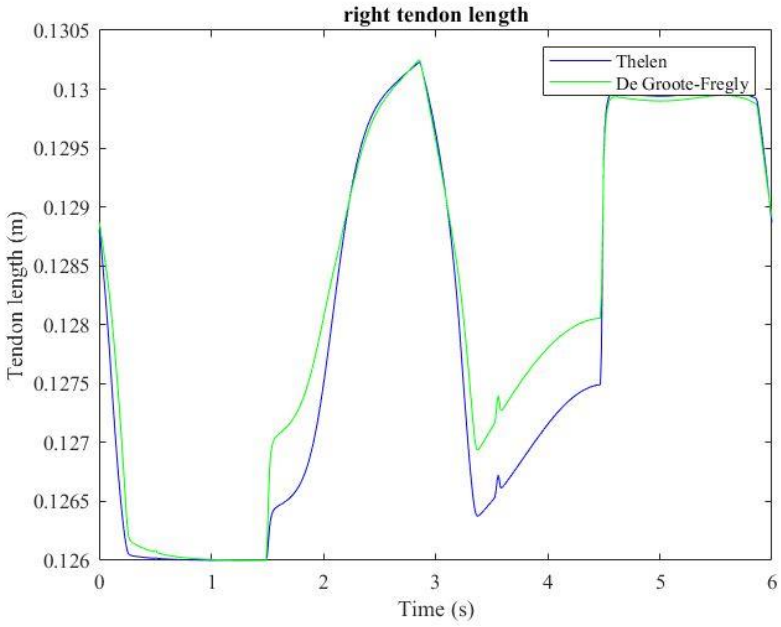


Figure 9.3. Normalized tendon length in right muscle. OpenSim Moco [13]

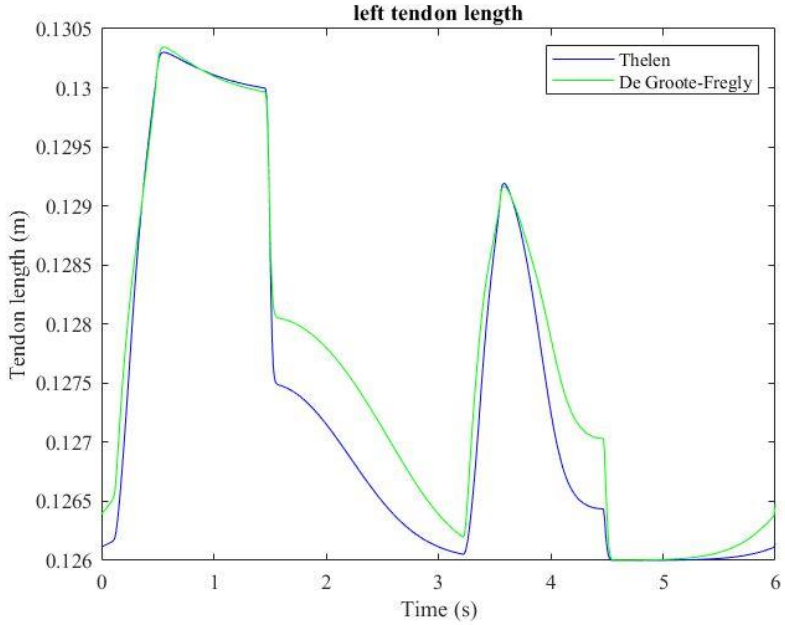


Figure 9.4. Normalized tendon force in left muscle. OpenSim Moco [13]

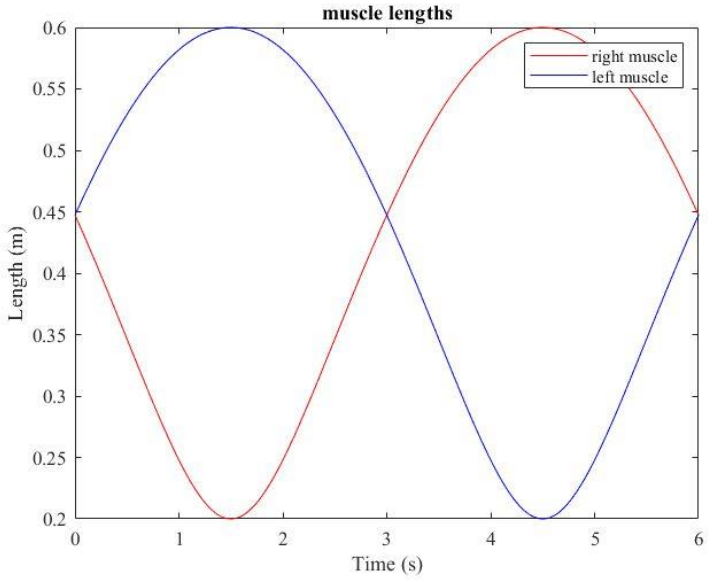


Figure 9.5. Muscle lengths. OpenSim Moco [13]

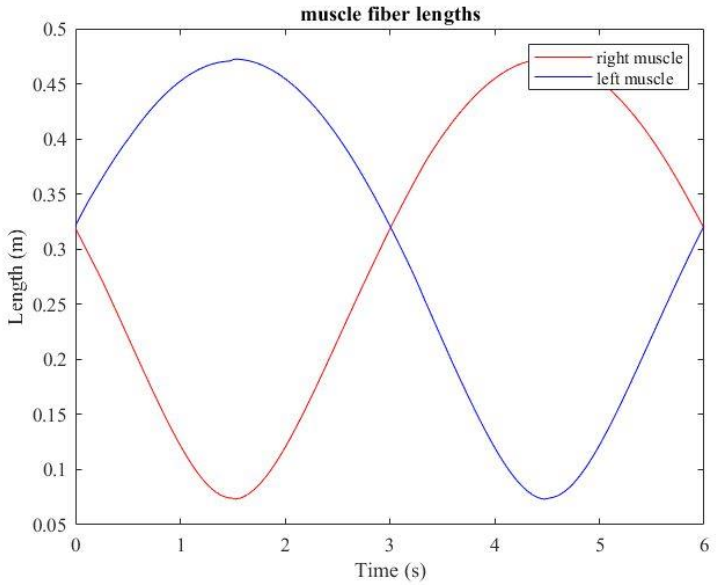


Figure 9.6. Muscle fiber lengths. OpenSim Moco [13]

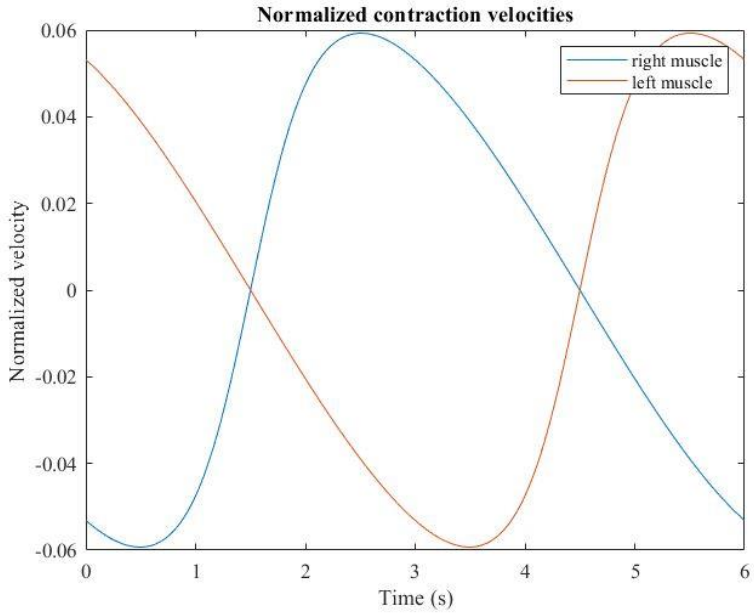


Figure 9.7. Normalized contraction velocities in both muscles. OpenSim Moco [13]

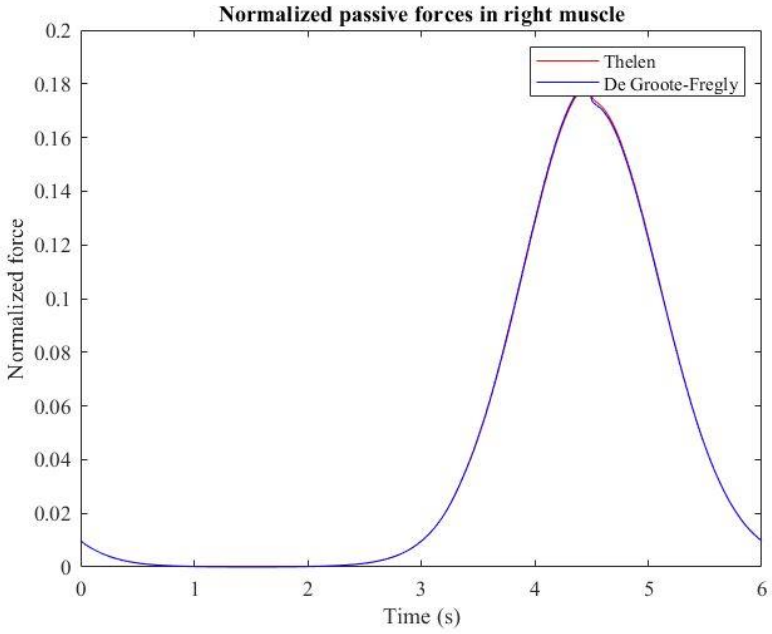


Figure 9.8. Normalized passive force in both models, right muscle. OpenSim Moco [13]

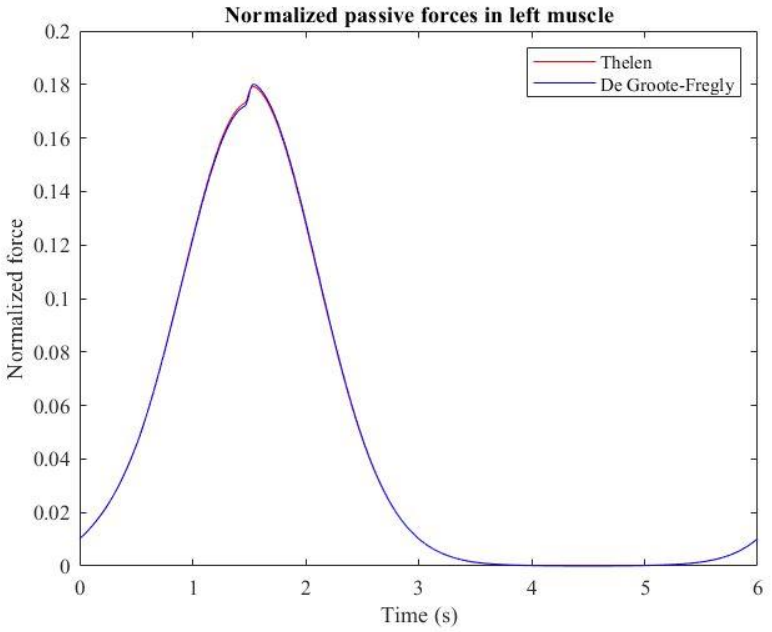


Figure 9.9. Normalized left passive force in both models. OpenSim Moco [13]

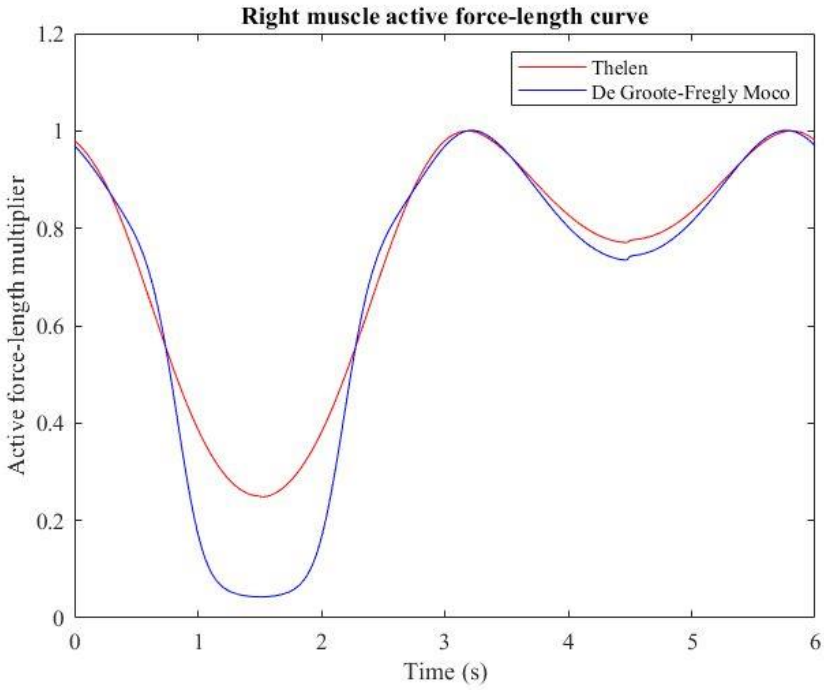


Figure 9.10. Right active force-length multiplier in both models. OpenSim Moco [13]

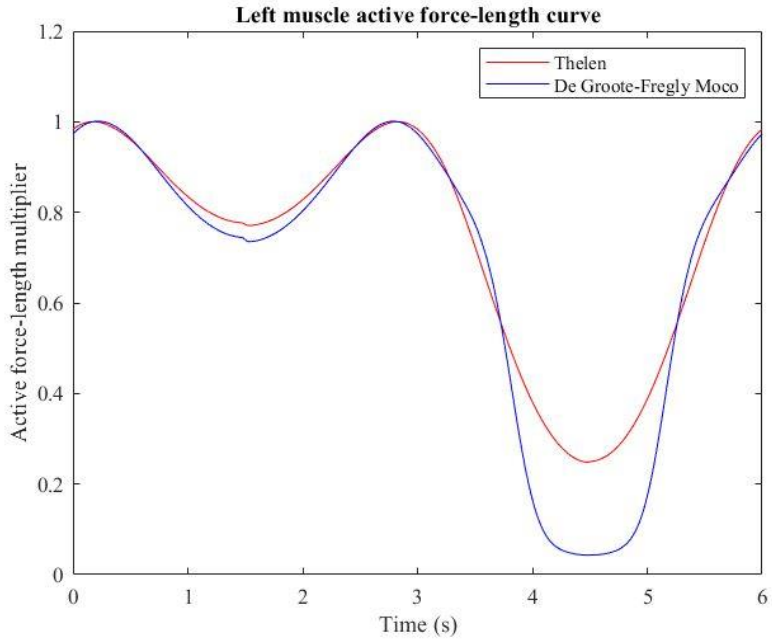


Figure 9.11. Left active force-length multiplier in both models. OpenSim Moco [13]

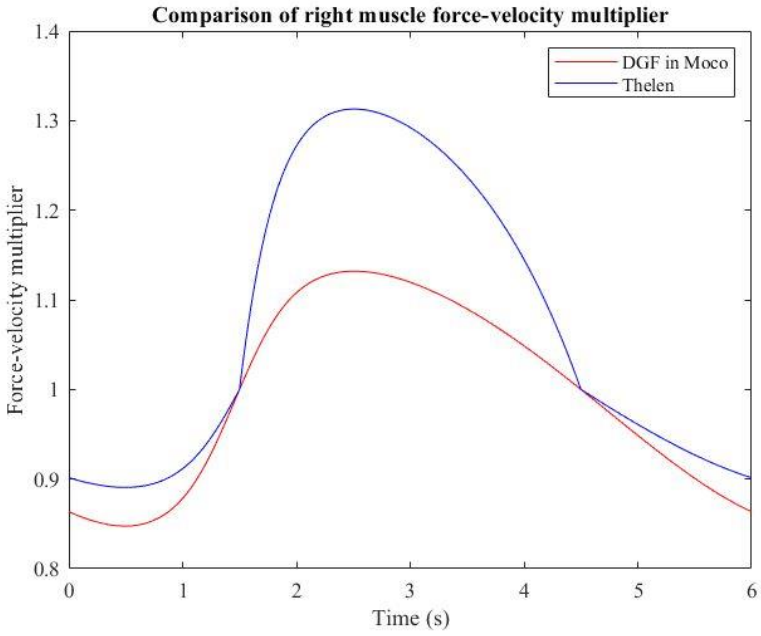


Figure 9.12. Right active force-velocity multiplier in both models. OpenSim Moco [13]

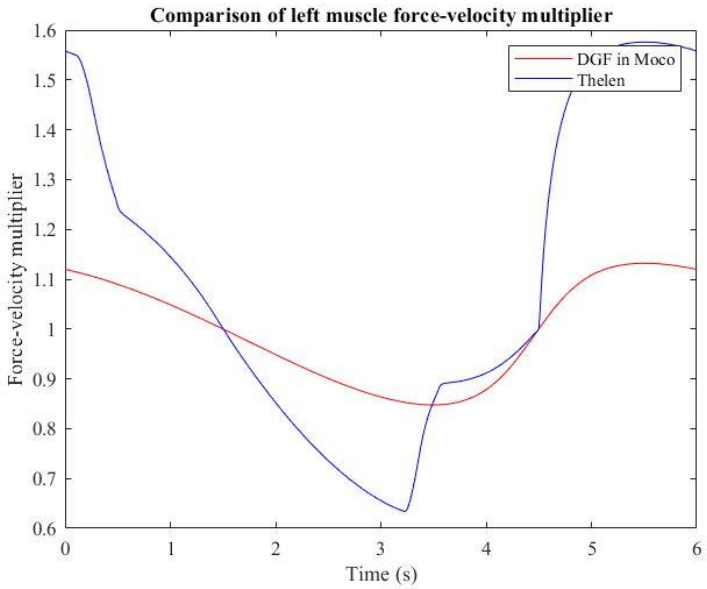


Figure 9.13. Left active force-velocity multiplier in both models. OpenSim Moco [13]

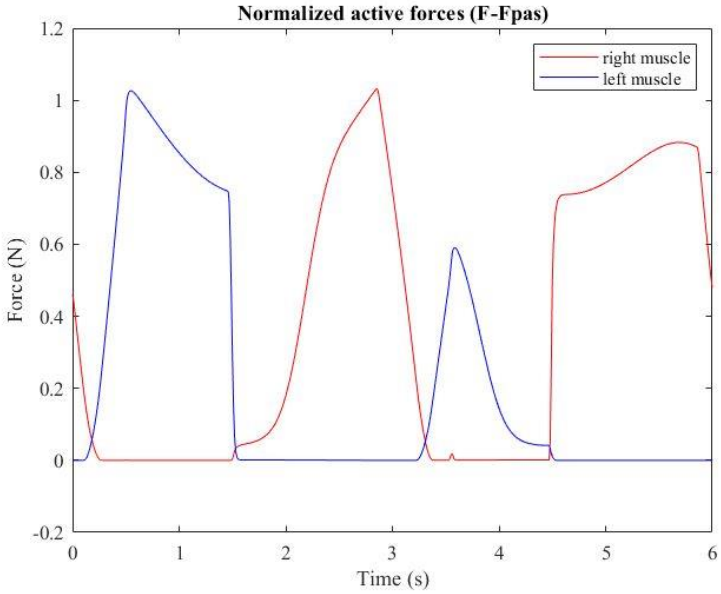


Figure 9.14. Normalized active forces in both muscles. OpenSim Moco [13]